UNIVERSITY OF BREMEN
INSTITUTE FOR
ARTIFICIAL INTELLIGENCE

Fast Robot Learning using Prospection and Experimental Knowledge

A Cognitive Approach with Narrative-Enabled Episodic Memories and Symbolic Knowledge

Asil Kaan Bozcuoğlu, M.Sc.

Vollständiger Abdruck der vom Fachbereich 3 (Mathematik und Informatik) der Universität Bremen zur Erlangung des akademischen Grades eines

Doktor-Ingenieur (Dr. -Ing.)

genehmigten Dissertation.

1. Prüfer:	Prof. Dr. h.c. Michael Beetz, Ph.D. *Universität Bremen*
2. Prüfer:	Prof. Dr. Kei Okada *The University of Tokyo*

Die Dissertation wurde am 28.02.2019 bei der Universität Bremen eingereicht und durch den Prüfungsausschuss am 06.05.2019 angenommen.

Abstract

Humans employ data-efficient learning mechanisms to obtain new skills and to improve the existing ones. Robots have already replaced humans in terms of labor in performing repetitive and dangerous tasks in structured environments like factories. On the other hand, developing completely autonomous robotic systems for unstructured environments, like a household, is still a challenge for roboticists due to the infeasibility of programming every possible case and today's data-hungry machine learning approaches. In order to assist humans in such environments, I believe that robots should be able to gain and improve skills using human-like learning mechanisms regularly.

For this purpose, I present a cognition-enabled fast learning framework in this dissertation which makes use of symbolic knowledge, episodic memories, and cloud robotics services along with a cutting-edge deep imitation learning methodology in order to reduce the dependency on big experiment data. Using this framework, robots can (1) **imitate** tasks demonstrated by a human demonstrator in virtual-reality, (2) **adapt** the actions of itself and others to new conditions, and (3) **prospect** which task parameters lead to the desired goal.

To validate these abilities, I have provided some experimental results. These experiments were conducted with four different service robots in various kitchen environments. The human demonstrations were recorded inside a game-with-a-purpose using virtual-reality equipment. Such a setup enables roboticists to crowdsource their training data by eliminating the requirement of being in the same environment with the robot.

Zusammenfassung

Menschen setzen dateneffiziente Lernmechanismen ein, um neue Fähigkeiten zu erwerben und die vorhandenen zu verbessern. Der Mensch wurde bereits durch Roboter ersetzt, um repetitive und gefährliche Aufgaben in strukturierten Umgebungen wie Fabriken auszuführen. Andererseits ist die Entwicklung vollständig autonomer Robotersysteme für unstrukturierte Umgebungen, wie beispielsweise ein Haushalt, nach wie vor eine Herausforderung für die Robotik. Zum einen ist es nicht möglich, jeden möglichen Fall zu programmieren und zum anderen bedürfen entsprechende maschinelle Lernverfahren große Datenmengen. Um Menschen in solchen Umgebungen zu unterstützen, sollten sowohl Roboter als auch Menschen neue Fähigkeiten erwerben und die vorhandenen verbessern.

Vor diesem Hintergrund befasst sich die vorliegende Arbeit mit einem schnell lernenden Rahmenwerk, das symbolisches Wissen, ein episodisches Gedächtnis sowie Cloud-Robotik-Dienste mit einer neuartigen Deep-Learning-Methodik kombiniert, um die Abhängigkeit von großen Experimentdaten zu reduzieren. Mithilfe dieses Rahmenwerks können Roboter (1) von einem Menschen in der virtuellen Realität demonstrierte Aktionen nachahmen, (2) die Aktionen von sich selbst und anderen an neue Bedingungen anpassen und (3) simulieren, welche Bewegungsparameter zum gewünschten Ziel führen.

Um diese Eigenschaften zu validieren, werden die durchgeführten experimentellen Ergebnisse vorgestellt. Diese Experimente wurden mit vier verschiedenen Servicerobotern in verschiedenen Küchenumgebungen durchgeführt. Die Demonstrationen wurden von Menschen in einem seriösen Spiel (Game with a purpose) in der virtuellen Realität aufgezeichnet. Dies gibt dem Forscher die Möglichkeit, das Sammeln der Trainingsdaten an eine Crowd auszulagern, da es bei dieser Vorgehensweise nicht erforderlich, dass die Trainingsdaten in der physischen Umgebung des Roboters gesammelt werden.

Acknowledgments

I would like to express my deepest gratitude to my *doktorvater*, Prof. Michael Beetz, for giving me the opportunity to work under his supervision during my doctoral studies. Being a member of Institute for Artificial Intelligence (IAI) was a great experience with the latest technology robotics equipment and the leading-edge research projects.

I want to thank Prof. Kei Okada for being my co-supervisor and supervising my internship in JSK Laboratory at the University of Tokyo. The research output we have produced together is a huge part of this dissertation.

I would also like to thank all of my colleagues, especially, Prof. Byoung-Tak Zhang for his mentorship during my Seoul visit, Daniel Beßler for being an awesome officemate and a co-researcher, and, also, for his valuable comments on this dissertation; Gayane Kazhoyan for organizing the Bremen part of the IAI-JSK collaboration, exchanging research ideas, and making suggestions for this dissertation; Yuki Furuta for his co-research during our collaboration; Seungjae Jung and Joonho Kim for the cooperation within GenKo Project; Andrei Haidu for helping me out in the virtual-reality aspects of this dissertation, Alexis Maldonado for his endless technical support, and Fereshta Yazdani for her co-research and organizational efforts in SHERPA project. I am also thankful to Moritz Tenorth for his mentorship at the beginning of my doctoral studies. And, of course, I gratefully acknowledge the management team of IAI, Hagen Langer, Sabine Veit, Lena Jacobs, Andrea Cowley, Insa Warms-Cangalovic, and Verena Engelhardt. The institute rests on your shoulders, and we cannot achieve these results without your help.

I would like to send my special thanks to my family. Mom, little sis and dad... without your love and support, I couldn't accomplish this far. *Aramızda kilometreler olduğunda bile desteğinizle hep yanımdasınız. İyiki varsınız! Kocaman teşekkürler!*

Last but not least, I want to thank love of my life, Duygu Yıldırım, for being always there for me and standing up to difficult times with me together! *Kar tanem, bi tanem, çok sağol*!

This work has received funding from DFG Collaborative Research Center 1320, EASE, and European Framework 7 Projects, RoboHow.Cog (Grant agreement ID: 288533), SHERPA (Grant agreement ID: 600958), and ACAT (Grant agreement ID: 600578).

Bremen, February 28th 2019 Asil Kaan Bozcuoğlu

Science is the most reliable guide for civilization, for life, for success in the world.
Searching a guide other than the science is absurdity, ignorance and heresy.

Mustafa Kemal Atatürk
The Founder of Modern Turkish Republic

Contents

List of Used Abbreviations and Symbols

Abbreviations

ACT-R ..... Adaptive Control of Thought-Rational

API ..... Application Programming Interface

BC ..... Behavioral Cloning

BURLAP ..... Brown-UMBC Reinforcement Learning and Planning

CRAM ..... Cognitive Robot Abstract Machine

DoF ..... Degree of Freedom

EASE ..... A Project: Everday Science and Engineering

FOON ..... The Functional Object-Oriented Network

GMM ..... Gaussian Mixture Model

HSR ..... Toyota Human Support Robot

HTN ..... Hierarchical Task Network

IAI ..... Institute for Artificial Intelligence at University of Bremen

IRL ..... Inverse Reinforcement Learning

MAML ..... Model Agnostic Meta-Learning

MDP ..... Markov Decision Processes

MGM ..... Multivariate Gaussian Model

NEEM ..... Narrative-Enabled Episodic Memory

NoSQL ..... Not Only Structured Query Language

OWL ..... W3C Web Ontology Language

PbD ..... Programming by Demonstration

PDDL ..... Planning Domain Definition Language

PR1 ..... Stanford University Personal Robot 1

PR2 ..... Willow Garage Personal Robot 2

RL ..... Reinforcement Learning

ROS ..... Robot Operating System

RPL ..... Robot Programming Language

SARSA ..... State-Action-Reward-State-Action

SHERPA ..... A Project: Smart collaboration between humans and ground-aerial Robots for improving rescuing activities in Alpine environments

SRC ..... Structured Reactive Controllers

SRDL ..... Semantic Robot Description Language

SQL ..... Structured Query Language

VR ..... Virtual Reality

w.r.t ..... with respect to

Symbols

b ..... bias

C_i ..... covariance matrix of a multivariate Gaussian model

d^h ..... human demonstrations

d^r ..... robot executions

i ..... the index of the cluster

n_i ..... the size of the cluster

W ..... weight matrix

x ..... the output of hidden layer's activation function

$\overline{X}_i$ ..... the mean of clusters in the working memory

y ..... next layer

z ..... the parameter vector concatenated to an output of hidden layer's activation function

ϕ_T ..... meta-parameter draft

$\mathcal{L}$ ..... loss function

θ ..... initialization parameter

ψ ..... adaptation parameter

List of Figures

List of Tables

List of Algorithms

Chapter one

Introduction

The word *robot*, which has been derived from the Czech word *robota* (*Eng*: forced labor), was coined by K. Čapek in his play R.U.R. "Rossum's Universal Robots" (1920) to denote a fictional humanoid agent. After the advancements in electronics and mechanics in the 20^{th} century, we have seen real robots and, even, witnessed that they have replaced humans in terms of labor in performing repetitive and dangerous tasks in structured environments like factories.

With the latest advancements in technology, robots are envisioned to assist humans in overcoming everyday tasks. Such assistance will, primarily, improve the quality of life among elderly and disabled people. Towards this goal, we have already seen a variety of service robots such as Willow Garage Personal Robot 2 (2011) and Care-O-bot 3 (2011) from Fraunhofer that are physically capable of human-level manipulation tasks. In early studies and demonstrations, we have witnessed that such robots do complex manipulations with purely teleoperation techniques (Wyrobek et al., 2008) or with limited autonomy (Okada et al., 2003). Furthermore, Willow Garage and some research laboratories have made impressive demonstrations showing that PR2 is capable of service tasks such as serving a beer[1], fetching a sandwich from a restaurant[2] and preparing breakfast[3].

Although these show that the hardware is proven to be sufficient for such service tasks, the robots are still not serving us in everyday life due to lack of long-term autonomy. The infeasibility of programming every possible case is the main limitation of the

[1] https://youtu.be/c3Cq0sy4TBs
[2] https://youtu.be/RIYRQC2iBp0
[3] https://youtu.be/_SIUCrmE8J0

realization of robots in people's home. Learning is a promising alternative; however, traditional learning methods often need large sets of training data and supervised training which requires time and technical expertise. A leap-forward at this point would be combining different learning methods and sources in a heterogeneous way.

Along with this idea, this dissertation presents a learning framework that combines imitation learning, learning from other robots, tasks, and environments. The rest of this chapter is organized as follows. I begin with my motivation for writing this dissertation. Then, I formally define the problem of interest. After that, I briefly describe the research scenarios that I have based the experiments in this dissertation. Later, I overview the requirements for the implementation of the presented framework. I conclude the chapter with the contributions of this work.

1.1 Motivation

Deep learning-based techniques start to give promising results in the domain of machine learning thanks to modern graphics processing units' parallel processing abilities. Roboticists have demonstrated empirical success in diverse applications such as robot vision (Eitel et al., 2015), grasp learning (Pinto and Gupta, 2016), motion control (Zhang et al., 2015), and human interaction (Baccouche et al., 2011).

Large-scale data is often required to train these systems. Such a mandate undoubtedly hurts the scalability as the training data needed exponentially grows while adapting to different variances. For instance, Figure 1.1 depicts Google's setup in which their robot arms are run for long hours to collect a dataset for deep grasp learning applications. Although it might seem like a practical solution to gather up training data this way, dedicating robots for long-lasting random experiments is, nevertheless, a challenge for many research laboratories and consumers. First, accessing robot hardware is still a challenge in today's world. Even researchers that have access to robot hardware need to share it with other lab members. That means occupying it for a long time can make fellow researchers unhappy. Similarly, generating some portion of data in simulation does not solve this challenge completely because roboticists need to put a lot of programming effort to tweak their control executives to have inherent noise and variances in a simulation environment.

On the other hand, humans are better skill learners in terms of **data efficiency**. Starting from the early infancy, humans grasp new skills and improve the existing ones to sustain their lives. Instead of purely relying on demonstrations or instructions, they

Figure 1.1: *Google's setup for the generation of large-scale robotic grasp data.*

employ additional cognitive abilities such as forming high-level concepts, applying logic, and reasoning (Gardner, 2011).

Throughout my doctoral studies, I have developed similar **data-efficient learning** mechanisms that ease training efforts requiring time and technical expertise. In the rest of this dissertation, I have referred to these mechanisms as **fast learning** abilities since they substantially decrease data gathering and training phases. I believe that human learning is a perfect role model to this end. Thus, cognitive psychology studies modeling human learning guide my research towards such abilities.

1.2 Problem Definition

Recently, roboticists begin to implement robots for performing human-scale manipulations, such as making pancakes (Beetz et al., 2011b), conducting chemical experiments (Lisca et al., 2015), using a dishwasher (Contreras et al., 2018), and cleaning up a room (Hollerbach et al., 2009) (see Figure 1.2). Although these robots are major milestones towards bringing robots to our daily life, they have still

Figure 1.2: *The robotic manipulations mentioned in Section 1.2. (a) Beetz et al. (2011b)'s pancake making setup, (b) Lisca et al. (2015)'s robotic experiment for the Ocean Sampling Day (c) Contreras et al. (2018)'s HSR opening a dishwasher at Robocup@Home, and (d) Hollerbach et al. (2009)'s PR1 executing tidying-up-a-room.*

many limitations which prevent them to be integrated into everyday life. In my doctoral studies, I have focussed on two of these limitations which are as follows. Firstly, these skills are implemented by considering environmental and context-related properties. Such implementations imply that moving these robots into different environments or changing the existing environment will require reprogramming or retraining (**Limitation 1**). Secondly, these robots still operate with a higher level of failures compared to humans. Even a slight misplacement in the environment or off-parameterization can cause the whole execution to fail (**Limitation 2**).

In order to tackle these limitations, different approaches can be taken. The ones that I particularly find promising and exciting are as follows (also illustrated in Figure 1.3). In the case of **Limitation 1**, enabling robots to increase their action repertoire by imitating others with only a few demonstrations (denoted as **Approach 1A**) would substantially reduce the overhead caused by reprogramming and retraining. Moreover, providing geometrical and capability reasoning mechanisms for

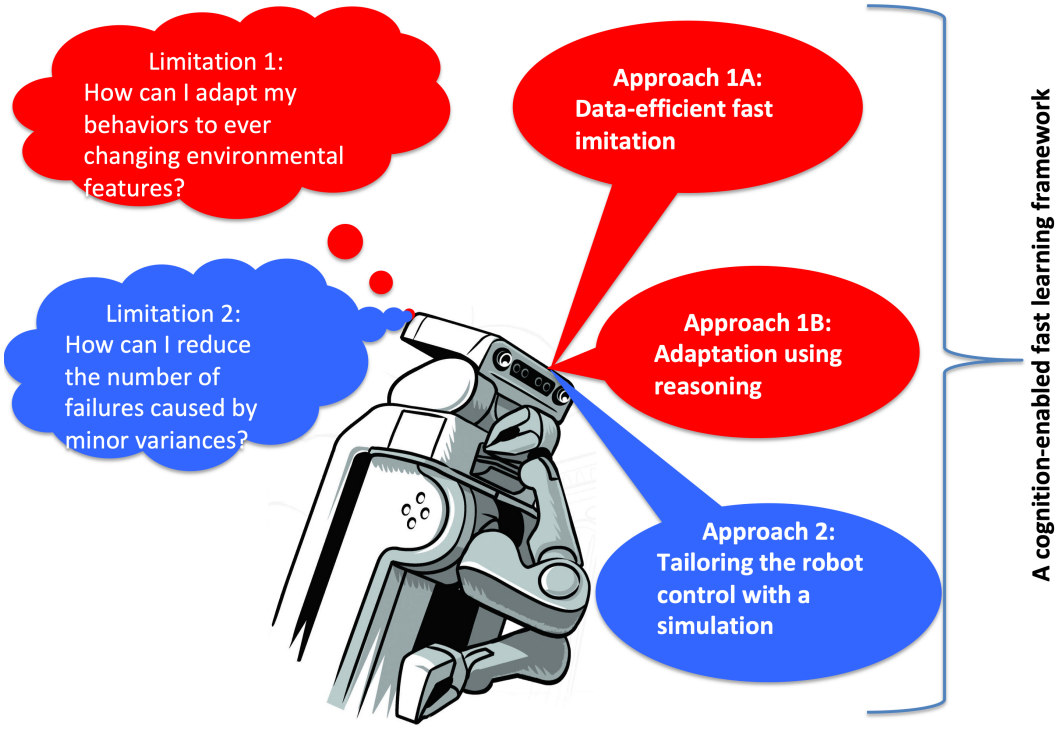


Figure 1.3: *The approaches taken in this dissertation in order to tackle with* ***Limitations 1 & 2****.*

adapting behaviors to new environments (**Approach 1B**) can similarly help to advance towards **Limitation 1**. In order to reduce the failures explained in **Limitation 2**, a worth-investigating approach is perceiving the world state (at least partially) and anticipating what can go wrong and how that can be avoided in a fine-grained resolution. This idea can be implemented by coupling a physics-enabled simulation with the robot's control and perception executives (**Approach 2**).

To sum up, **the problem of interest** in this dissertation boils down to how a **cognition-enabled fast learning framework** can be implemented for the realization of these approaches.

1.3 Research Scenarios

Similar approaches, namely **imitation learning**, **action generalization**, and **prospection**, are used by humans in everyday encounters. In order to validate the presented

research, I picked three of such encounters and designed robotic experiments based on them.

In infancy and early childhood, humans observe their caregivers and imitate their actions (Sommerville et al., 2005). These imitations become more significant when they are scaffolded or play games with their caregivers. As a similar scenario, a Toyota Human Support Robot (HSR) is scaffolded to execute very fundamental motion primitives, such as *reaching* and *pushing* in a kitchen environment(**Use Case 1**). The expected outcome of this scaffolding is that it can imitate the overall course of action even after a single **virtual-reality** (VR) demonstration (the validation of **Approach 1A**).

Moreover, humans can quickly adapt their skills to different circumstances (Gluck et al., 2011). Consider a child that wants to open an oven (**Use Case 2**). If she knows how to open a fridge, it is intuitive for her to generalize this action to the oven since both actions are intrinsically opening a container with a hinged joint door. Such positive influences from other agents would also substantially increase the effectiveness of robots in unstructured environments without reprogramming or retraining. I use **Use Case 2** as the testbed for **Approach 1B**. By witnessing an *opening-a-fridge* demonstration performed by a different robot, the target robot is expected to adapt the parameters to another task, *opening-a-oven*, using reasoning. The expected outcome is being able to infer two actions as the same and to categorize as *opening-a-container-with-hinged-joint-door*.

Finally, humans are very good at predicting the results of their actions. They intuitively **prospect** to infer the outcomes of complex tasks just before the execution. This ability helps them to fine-tune their actions with respect to factors such as speed, positioning, and applied force (Druckman et al., 1992). One can consider an encounter where a person wants to reach a ketchup bottle beyond his reach. Before trying to achieve, he simulates reaching in mind and decides a trajectory that may achieve his goal and satisfy other task constraints. Similar to this, I test the realization of **Approach 2** with **Use Case 3** in which a Personal Robot 2 (PR2) is expected to optimize its reaching motion using reinforcement learning setup in a simulation integrated with its control executive. Thanks to its integration with the symbolic knowledge base, it is also possible to compare the prospected trajectories with respect to distance and duration with predicate logic queries.

1.4 Requirements

In order to realize **Approaches 1A, 1B, & 2**, some narrative forms, in terms of knowledge and memory types, should be available to the robot. In this section, I name these narratives and explain their usage in the presented framework.

1.4.1 The Need for Context-Specific Knowledge and Self Knowledge

Context-specific knowledge is a type of knowledge that relates to conceptual facts, information, and descriptions of a particular domain or an environment. It enables agents to understand, reason, and act intelligently in that domain. Humans acquire this knowledge by synthesizing various sources such as observations and interactions with others starting from the early childhood period (Perkins and Salomon, 1989; Phillips et al., 2000).

Fulfilling even a simple household task requires a rich set of context-specific knowledge and the ability to perform fine-tuned actions. If a robot is delegated a "bringing milk bottle to the table" task, it must have some knowledge on the kitchen domain to achieve this task. For instance, it needs to know where to look for a milk-box instead of searching in a brute-force manner due to time constraints. Thus, the facts that milk is a perishable liquid and that perishable items are usually stored inside a fridge should be available in the knowledge base. The robot should similarly know the fridge's location in the kitchen.

The term *self-knowledge* denotes the knowledge about the agent's physical properties, sensing, and acting capabilities in robotics. This knowledge is particularly important while imitating the other agents. Consider a robot to open a fridge which was opened by another robot earlier. Using available self-knowledge and the execution log of this run, it can infer whether its arm is capable of executing a similar opening trajectory.

Although there are alternative ways to make these two knowledge types available to the robot control programs, perhaps the most proper way would be the automated acquisition of knowledge by the robots themselves with long-term autonomy and life-long learning like humans. However, due to the limitations in long-term autonomous platforms as stated by Jagielska et al. (1999) and Argall et al. (2009), and, also, not being the focus of this work, I mainly rely on static ontologies in the form of *encyclopedic knowledge* for supplying robots context-specific knowledge and self-knowledge.

1.4.2 The Need for Episodic Memories

Intelligent robots would profit immensely from a memory system that enables them to remember what they have done, why, how, and what happened. In comparison, human memory is often categorized into two subsystems: Short-term and long-term memory (James, 1890). The former is a low-capacity store that provides the continuum for accomplishing the present tasks. The latter is a high-volume store that provides detailed information for all kinds of tasks. Wood et al. (2012) further categorize human memory along with other directions, such as procedural versus declarative memories, where the procedural memory contains subconscious or compiled information and the declarative memory is considered as a conscious available recollection of factual information and previous experiences. **Episodic memory**, a subcategory of the declarative memory, stores experienced event information that is temporally and spatially sorted and attached with context information. Humans use their episodic memory for improving their manipulation skills. In other words, they recall their past executions for reasoning on what/why/how they did things and investigating ways to improve them (Tulving, 2002). In a way, one can consider humans as *self-aware* and *self-correcting* agents that will enhance their abilities over time.

In order to function similarly, episodic memories from a robot performing complex manipulation tasks should contain low-level data, such as images from perception routines that influenced decision making, parameterization of complex motions, and effects of these motions. Besides, these memories should include high-level representations of the intended actions and the belief-dependent decisions that led to the chosen course of action. As an example, a "breakfast preparation" episodic memory should not include only motion data but also the information of what the ultimate goal of "opening the fridge" subtask is. Furthermore, different than humans, robots produce these memories as digital entities. Having such a rich knowledge of past experiences makes it possible to use these memories as mediators during the knowledge exchange and skill transfers.

As robot perception is out of this dissertation's scope, the presented framework uses episodic memories not only for keeping the record of robots' own experiences but also accepting the human demonstrations and experiences of other robots. For this manner, *narrative-enabled episodic memory (NEEM)* (Winkler et al., 2014; Beetz et al., 2018) which can store symbolic and subsymbolic information about task executions is used. The symbolic part includes the task information recorded by the planners and the force-dynamic event information coming from the game engines (in the case of virtual reality demonstrations). In the subsymbolic part, the joint states, trajectories,

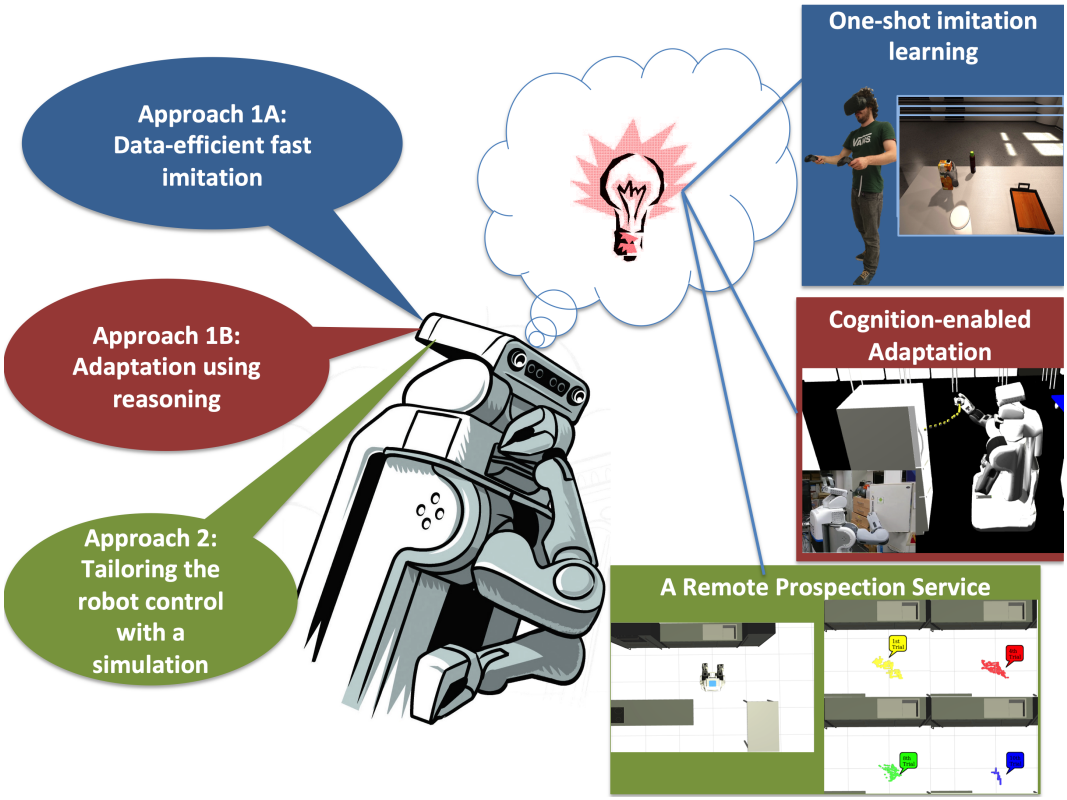


Figure 1.4: *The contributions of this dissertation in order the service robots to gain the ability of intelligent manipulation ability.*

sensory data and camera images are available. This information is aligned with the rest of the robot knowledge base called KnowRob (Tenorth and Beetz, 2013; Beetz et al., 2018). By having such "diaries," the robot would have crucial insights into the actions performed by the demonstrator. NEEMs are further explained in Section 2.3.

1.5 Contributions

This dissertation reports my research on implementing **Approaches 1A, 1B & 2** inside a fast-learning framework (Figure 1.4). With this, I contribute the state-of-the-art in the following directions.

Imitation learning is, in a nutshell, a machine learning type which trains a policy based on demonstrations to perform similar actions. As deep learning advances, the roboticists come up with hybrid techniques which are sometimes called *few-shot*

imitation learning. Although these techniques undoubtedly contribute in the direction of human-level fast learning, there is still a gap between humans and robots in the sense that the robots are not mainstream consumer products. Thus, it is not currently possible to scaffold robots with an ordinary human "caregivers". In order to lower this gap and realize **Approach 1A**, I push this domain further by making use of virtual-reality demonstrations in a *few-shot imitation learning* setting. Such virtual reality demonstrations enable humans to teach robots new skills without the necessity of sharing the same environment.

Limitation 1 states that most of today's service robotics tasks require environment- and agent-specific implementations. Although the high-level structure of these tasks mostly stays the same, the motion parameterizations are highly dependent on the environmental and agent-specific properties. Thus, a generic implementation that succeeds most of the cases is not possible with today's technology. In my doctoral studies, I contribute to this research area by integrating a reasoning mechanism on top of execution logs, semantic maps, and robot descriptions. The robots can reason how motion parameters need to be adapted to the new circumstances using the presented semantic rules (the realization of **Approach 1B**).

Due to the complex and non-reversible[2] structure of household environments, it is essential for the service robots to operate with lower failure rates and not to create undesired effects. For this manner, this dissertation offers a prospection service that enables robots to find out a set of motion parameters that leads to the desired goal using reinforcement learning (the realization of **Approach 2**).

1.5.1 Approach 1A: Fast Imitation Learning for Robots

In order to have learning rates close to humans' during encounters such as **Use Case 1**, a fast and practical imitation learning mechanism should be available for service robots. As mentioned previously, humans achieve this by developing a **meta-manipulation ability** and adapting this ability to newly demonstrated actions.

In recent years, *few-shot imitation learning* techniques give promising results in terms of teaching actions even after processing a small number of demonstrations. Among those, one-shot imitation learning ones are especially practical for intelligent manipulation because of increasing manipulation skills by providing fast learning rates and transferring skills from human demonstrations. These approaches contain

[2] What I mean by non-reversibility is that manipulations that change the state of the world cannot be undone with ease by the robot.

an intermediate learning phase (Finn et al., 2017b; Yu et al., 2018). Whenever a new action is demonstrated, the meta-model is adapted using gradient updates.

In this work, I extend one of these techniques called *domain-adaptive meta-learning* by Yu et al. (2018) so that it can be trained using NEEMs recorded in virtual reality. Using VR demonstrations in imitation learning has two significant advantages compared to the tracking- or perception-based demonstrations. First, it is possible to develop serious games for data collection where players can demonstrate tasks in a virtual environment without the necessity of sharing the same environment with robots as virtual reality games can provide photorealistic and physics-enabled environments. Second, roboticists have access to the ground truth of these demonstrations via the game engine. As aforementioned, this dissertation does not focus on the perceptual aspects of demonstrations. Instead, the robots are provided with the demonstrations of other agents with annotations and descriptions from either control executive or game engine as "digital diaries" in the format of NEEMs. Using the ground truth information, NEEMs can be generated automatically without manual labeling or relying on pattern recognition/machine learning techniques. The fast imitation learning methodology using VR demonstrations is described in detail in Chapter 5.

1.5.2 Approach 1B: Adaptation of Robotic Actions

A robot control executive can be considered as a master of an action if it is proficient at executing that action despite environmental variances and even using different robotic embodiments. Since there are infinitely many cases to handle in unstructured environments, it is impossible to address all of them by programmers at the time of implementation. Thus, a more feasible approach is to enable the robot control executives to understand the essence of situations and action requirements and to adapt their actions accordingly. In order to achieve that, they need to have abstract semantic representations of actions together with *context-specific knowledge* about the environment and robotic embodiments.

The presented framework contains a similar approach which makes use of previous experiences of robots, semantic descriptions of robots, and semantic maps. By using existing semantic rules, the robot control executives can reason how they can adapt the past actions to different environments and different embodiments. This methodology is described in detail in Chapter 8.

1.5.3 Approach 2: Robotic Prospection

Robots that execute everyday manipulation tasks can hugely benefit from being able to predict the consequences of their actions just before the execution. A prospection mechanism tailored to the robot control executive would be helpful in terms of finding appropriate parameters and avoiding failures. However, such a physics-enabled simulation is usually computationally-expensive and may not be achieved with agents' self-computing facilities.

In this work, I present a remote prospection service inside openEASE cloud engine (Beetz et al., 2015). Researchers and robots can describe the world model, the state of the agent and the problem that is being dealt with using this service. In return, it simulates the world and runs a learning algorithm and suggests a solution to reach the desired effect. This service is described in detail in Chapter 6.

1.5.4 Validation

Each of these contributions is validated by conducting a set of experiments with four different service robots, namely two Willow Garage PR2s, a Fetch Robot, and a Toyota Human Support Robot (HSR) in two different kitchen environments.

In Chapter 5, I report the experiments conducted for the validation of **Approach 1A**. In these, I have also used a rendered version of these environments to record human demonstrations in virtual-reality.

Chapter 6 contains how **Approach 2** is realized and used as a cloud service by the service robots. In Section 6.4, a PR2 robot uses this service to find out how to traverse to a location and how to reach an object.

Chapter 8 presents the experiments in which two PR2s and one Fetch robot adapt others' actions for their setup. These experiments showcase the realization of **Approach 1B**.

Besides these main experiments, there also exists other supplementary experiments in Chapter 7 and Chapter 9 which examine intermediate research products and test additional capabilities.

The results are considered to validate the contributions if the success rate exceeds the state-of-the-art (**Approach 1A**), the robots successfully adapts the action (**Approach 1B**), or the presented service is integrated into the robot control loop (**Approach 2**).

1.6 Outline of This Dissertation

This dissertation is structured as follows:

Chapter 1 provides an introduction together with the motivation and background information for this work and summarizes this work's contributions to the field.

Part I

Chapter 2 presents the architecture of the presented framework and outlines the components, software frameworks, and technologies used in this framework. Moreover, it introduces the notion of *Episodic Memories*. Then, the format of episodic memories used in this dissertation together with the mechanisms for generating and using these memories are presented.

Chapter 3 gives the theoretical background information for learning in robotics and knowledge representation and reasoning. The literature survey of these subjects is also given in this chapter.

Chapter 4 defines how robot control executives can interface with the cloud platform and record and access episodic memories.

Part II

Chapter 5 presents the fast learning mechanism used for imitating actions demonstrated by humans in virtual reality (**fast imitation**).

Chapter 6 describes how robots can use the cloud-based prospection service in order to parametrize their motions before executing them (**robotic prospection**).

Part III

Chapter 7 provides an affordance modeling methodology that allows robots to generate affordance models for their actions.

Chapter 8 presents an experience exchange methodology that enables robots to generalize actions and to exchange experiences between them using episodic memories and the cloud platform (**action adaptation**).

Part IV

Chapter 9 presents a semantically-annotated video rendering process from episodic memories. These videos are used for the consistency-check of episodic memories concerning belief-state and sensory data.

Part V

Chapter 10 concludes the dissertation with an overall content summary, a lessons-learned and the planned future work.

Part I

Preliminaries and Literature

Chapter two

System Overview

This chapter, first, introduces the presented framework by revealing its component-level design. After that, the knowledge representation and processing tools used in the presented framework is introduced. Finally, the episodic memory formalism used is explained in detail.

2.1 Architecture

The implementation of this framework is mainly based on Robot Operating System (Quigley et al., 2009b) (in short ROS). Despite its name, ROS is not an operating system but a middleware which provides infrastructure for a heterogeneous software architecture by providing hardware abstraction, communication mechanisms between processes, package management, and low-level device control.

As Figure 2.1 shows, the presented framework consists of three layers, namely, **imitation**, **adaptation**, and **prospection**. Depending on its need, the robot may use one or more layers by sending predicate logic queries implemented in SWI-Prolog (Wielemaker, 2003). SWI-Prolog is one of the open-source implementations of a logic programming language, called Prolog, mainly for semantic web applications. It has a rich set of libraries for predicate logic programming and ontology processing. The documentation of SWI-Prolog is available in the website `http://www.swi-prolog.org/`.

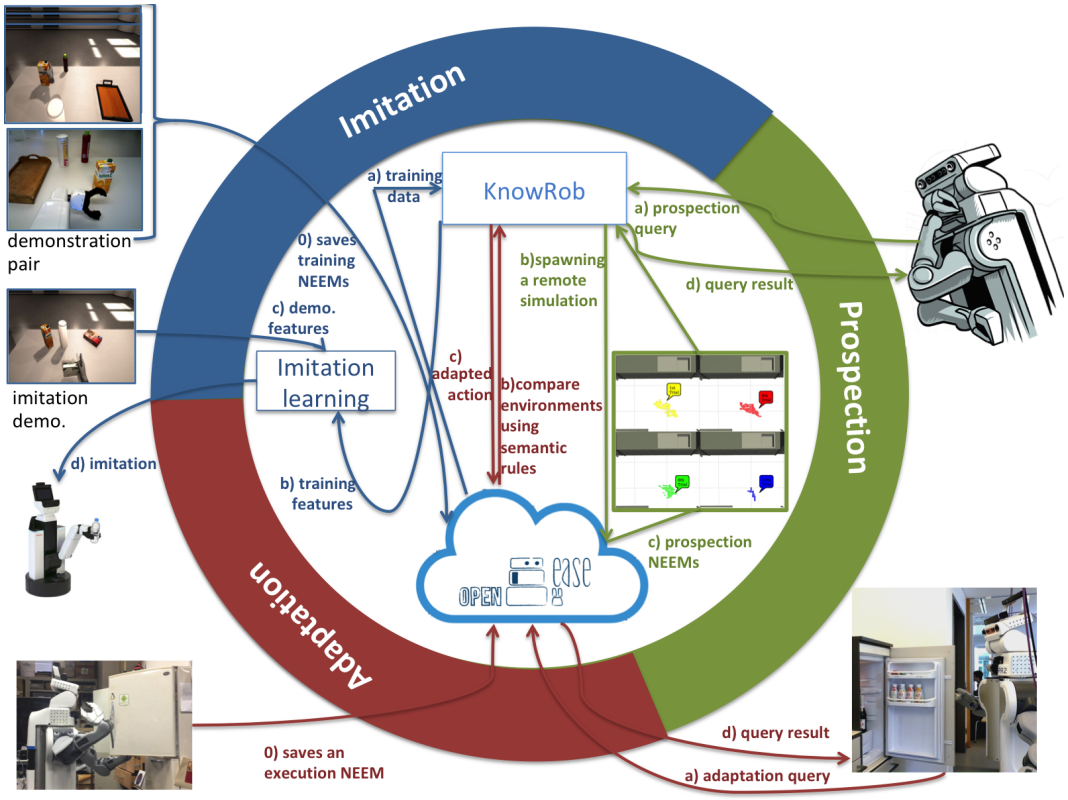


Figure 2.1: *The system architecture of the presented fast learning framework.*

Sending Prolog queries and receiving answers are achieved via ROS service calls, synchronous remote procedure calls. In ROS, components can offer a service under a string name, and a client calls the service by sending the request message including some parameters and awaiting the reply.

The inter-component communications except for the ones with openEASE (Beetz et al., 2015) are handled using such calls. In the case of passing high-volume data such as low-level training features, the sender sends this data inside a file. openEASE communication protocol is designed using WebSockets. This communication protocol is presented in Chapter 4.

2.1.1 Component Overview

In this subsection, I briefly overview components depicted in Figure 2.1. Each of these components is presented in detail later in this chapter.

Imitation Learning

This component is responsible for conducting the fast imitation learning methodology for learning manipulation skills from virtual-reality demonstrations. In the first phase, this component trains a learning algorithm called meta-learning using demonstration pairs which consist of n virtual reality demonstrations and a robot demonstration of the same task. In the second phase, the component imitates the task in the real world using a single virtual reality demonstration.

KnowRob

KnowRob (Tenorth and Beetz, 2013; Beetz et al., 2018) is a knowledge representation and reasoning tool for robots which contains static ontologies and dynamic knowledge sources in order to equip robots with context-specific knowledge for robot experiments. In the presented framework, KnowRob provides environmental knowledge, self-knowledge, predicates for reasoning and learning, and semantic rules for adaptation.

openEASE

openEASE is a cloud robotics platform which enables roboticists and robots to upload ontologies and their execution logs in the format of NEEM and to use the ones that were uploaded by others. In the presented framework, openEASE keeps NEEMs from virtual-reality demonstrations and robotic experiments and provide the knowledge inside these NEEMs when necessary and offers the remote prospection service.

2.1.2 Imitation Layer

For the realization of **Approach 1A**, the imitation layer is used. Figure 2.1 depicts this layer, its interaction arrows and its component, *Imitation Learning*, in dark blue color.

First, an initial training process, called meta-training, must be carried out by *Imitation Learning* using a set of demonstration pairs which are stored inside openEASE cloud

platform (see Section 2.2.2). Each of these pairs includes *n* virtual-reality (VR) demonstrations and one robot execution of the corresponding training task. A unique setting of the same environment is used in every demonstration. For this manner, the game engine renders a photorealistic version of the real world. In this process, a mapping between the avatar in virtual reality and the robotic embodiment is learned.

Once meta-training is completed, the layer gains the ability of fast-imitation by acquiring only a single VR demonstration of the new task. This fast-learning methodology is explained in detail in Chapter 5.

2.1.3 Adaptation Layer

In order to reuse past action executions in new environments or to make use of others' executions, robots can make use of the adaptation layer (Figure 2.1/maroon color). For this manner, a *narrative-enabled episodic memory* (see Section 2.2.1) (in short NEEM) from a past execution should be available in openEASE.

The predicates offered by the adaptation layer apply the semantic rules inside KnowRob (see Section 2.2.2.1) which tailor the motion parameters and subsymbolic data like trajectories to the new circumstances using semantic maps of the source and target environment and semantic robot descriptions. During execution, the robot sends an adaptation query using these predicates. As a result, the adapted data is returned to the robot.

The adaptation layer together with the semantic rules and predicates is explained in more detail in Chapter 8.

2.1.4 Prospection Layer

This layer provides an interface to the remote simulation service which enables robots to prospect. A typical prospection query includes a semantic map to be loaded as the simulation environment, an action-taking problem (i.e., how the robot ought to take actions in the given environment), and a reinforcement learning setting (i.e., learning algorithm, parameters and award function).

After a prospection query is sent, KnowRob interfaces with openEASE to spawn a simulation with the desired setting. The simulation records a NEEM for each reinforcement learning iteration. Later, those NEEMs are used for finding out optimal

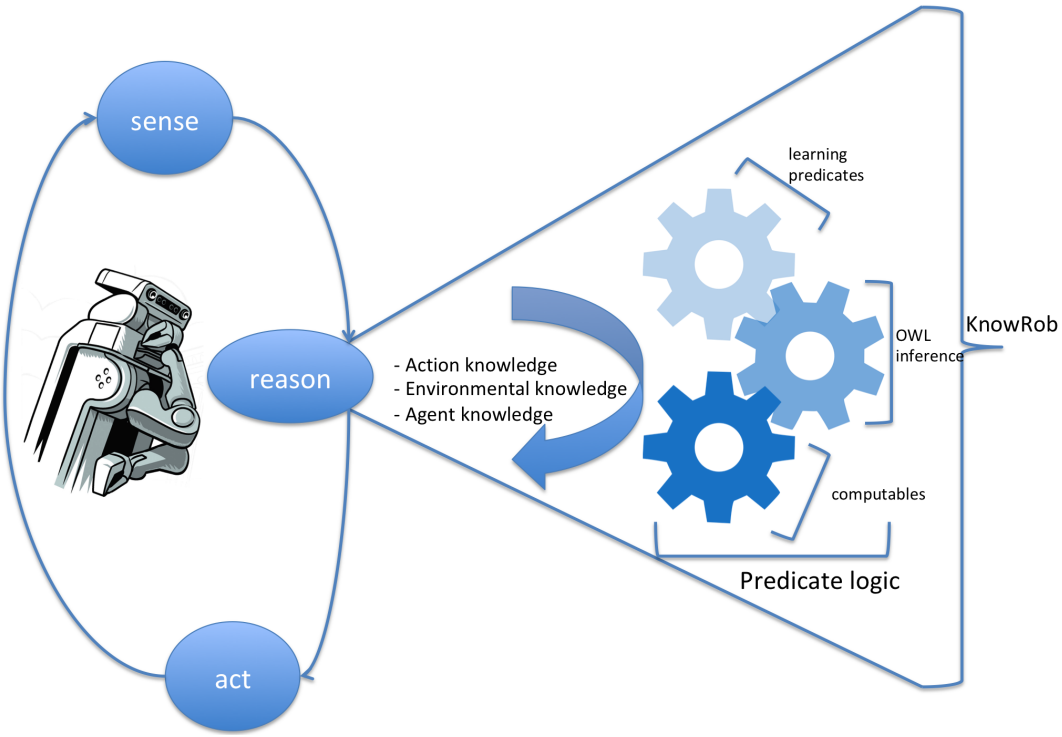


Figure 2.2: *KnowRob uses predicate logic implemented in SWI-Prolog (Wielemaker, 2003) in order to interface with different knowledge types available within itself. In the sense-reasoning-act loop (i.e. the robot control loop), agents can access this knowledge in their reasoning processes.*

actions that can be executed by the robot. More details on the prospection layer and the remote simulation service can be found in Chapter 6.

2.2 Knowledge Representation and Processing

2.2.1 KnowRob: Knowledge Processing for Robots

KnowRob as a Framework

KnowRob (Tenorth and Beetz, 2013; Beetz et al., 2018) offers knowledge representation and reasoning capabilities for robots and contains ready-to-use ontologies about

the household domain in the format of W3C Web Ontology Language (OWL)[1]. Roboticists can extend those capabilities and ontologies to other domains and applications as in (Yazdani et al., 2018). Using JSON_Prolog service, robots can access these capabilities and ontologies within Robot Operating System (ROS) middleware (Quigley et al., 2009a). By offering shared semantics within robot components or between robots, KnowRob makes it easier to use the knowledge acquired by different software components and to integrate information from different sources. In a way, these shared semantics act as the interlingua within the robot control loop.

More concisely, KnowRob provides knowledge about the environment, agents, actions, and previous experiments. As Figure 2.2 depicts, predicate logic is the crucial instrument for accessing this knowledge. Using available predicates dedicated for inference, learning and just-in-time computation (computables) implemented in SWI-Prolog (Wielemaker, 2003), agents can formalize their queries requesting specific knowledge.

JSON_Prolog service acts as a question-answering mechanism which finds solutions that resemble the knowledge needed by the components of the robot control loop. The queries formalized in SWI-Prolog syntax using available predicates are bound with knowledge using backpropagation algorithm (Wielemaker, 2003).

Environmental Knowledge and Object Models

Robots that execute complex manipulation actions in unstructured environments need well-detailed environmental information so that they can adapt their executions accordingly. For instance, robots should be able to answer the questions of *Where am I?* and *Where should I traverse to perform this action?* for every task including navigation requirements. Moreover, robots need environmental and object knowledge for the adaptation of their high-level plans and parametrizing their motion primitives. If a robot wants to grasp a drink from a fridge, it needs to know geometrical and physical properties of this fridge such as the shape of the door handle and its location to parametrize its grasping motion.

KnowRob stores environmental knowledge inside *semantic maps* which encode the types and relations between environments and objects together with 3D maps providing the scene geometry and the object locations with respect to a reference frame. Robots can directly feed these maps to their localization and navigation modules. Semantic maps contain well-detailed object information. First, a type definition

[1] https://www.w3.org/OWL/

which reflects the arrangement in KnowRob type taxonomy. Every class connects to related class with *subClass* relation. Hence, one visits more specific types while they reach through the leaves starting from the root type *Thing*. Secondly, there exists articulation information for the objects that consist of movable parts such as drawers and cupboards. Articulation information between object parts is explicated through joint relations. For example, there exists a hinged joint between a cupboard and its door. Besides, objects are annotated with their geometrical and physical properties such as height, width, length, and also shape information. For the objects with complex shapes, there exists a way to link their description in the semantic map with their 3D mesh data.

In this dissertation, environmental knowledge is densely used in Chapter 8 for adapting action data into new circumstances.

Agent Knowledge

In order to reduce the complexity, modern robot software is designed to be modular consisting of independent, interchangeable components, such that each one manages a particular aspect of the robot control. This modular approach, nevertheless, brings two significant overheads, namely coordination with shared data structures and high-volume communication between components.

Knowledge representation with a common vocabulary offers ways to reduce these overheads mainly by providing an interlingua between components and, also, between different agents. This interlingua could considerably change the way of robot control, robot interaction, robot programming, and multi-robot communication.

KnowRob offers such an interlingua by linking descriptions of robotic components, i.e., sensors, joints, links, actuators and control programs, via capabilities to actions in an ontology. Such ontologies are written in KnowRob's *Semantic Robot Description Language* (SRDL) (Kunze et al., 2011). The knowledge about components which, in the end, allows to infer the required components for performing a given action includes capabilities such as the abilities of 3D perception, manipulation and navigation and physical properties such as joint limits.

In Section 4.4, I present a field robotics use-case where agent knowledge is used for synchronizing the knowledge employed in different agents of a robotic search-and-rescue team.

Knowledge about Actions

Actions are specified in terms of their types which are arranged in a taxonomic structure (Tenorth et al., 2012), which provides a vocabulary for describing and implementing high-level task plans in KnowRob. According to their needs, roboticists can extend this hierarchy by deriving new action types according to their needs.

These action classes provide blueprints about actions including pre-actors, post-actors and capabilities needed. For instance, a robot should employ a 3D camera to execute a 3D perception task. Similarly, a cutting event requires a tool with a sharp edge. After a successful cutting action, multiple pieces of the same object should be obtained. Such hints help developers to develop high-level robot programs which formally define how to order a set of action primitives and to parametrize them in order to satisfy the given goals. In a *cutting-an-object* plan, the control executive can infer that the robot must execute *pick-an-object* motion primitive with *knife* as the object of interest.

2.2.2 openEASE: Knowledge in the Cloud

A cloud knowledge service, called openEASE (Figure 2.3), is presented by Beetz et al. (2015). openEASE acts as a remote knowledge representation and processing tool for robots equipped with standard KnowRob ontologies about actions, agents, and environments together with big activity data performed by various agents. Researchers can interact with openEASE via its web-interface equipped with an SWI-Prolog console for querying the knowledge base, a 3D-canvas to temporally project the world state and three different visualization areas for statistics, experiment images and high-level plan designators (Figure 2.4). Robot control programs interface with this remote knowledge service via WebSockets (Fette, 2011) using dedicated APIs (Bozcuoglu et al., 2018a).

The architecture of openEASE relies on the virtualization techniques of the Docker framework (Merkel, 2014) to create separate virtual storage and processing units called containers, for each user. openEASE comes with KnowRob containers pre-installed. Additionally, users have read access to a storage container that contains NEEMs along with read/write access to user-specific data containers where user-specific ontologies and execution logs reside.

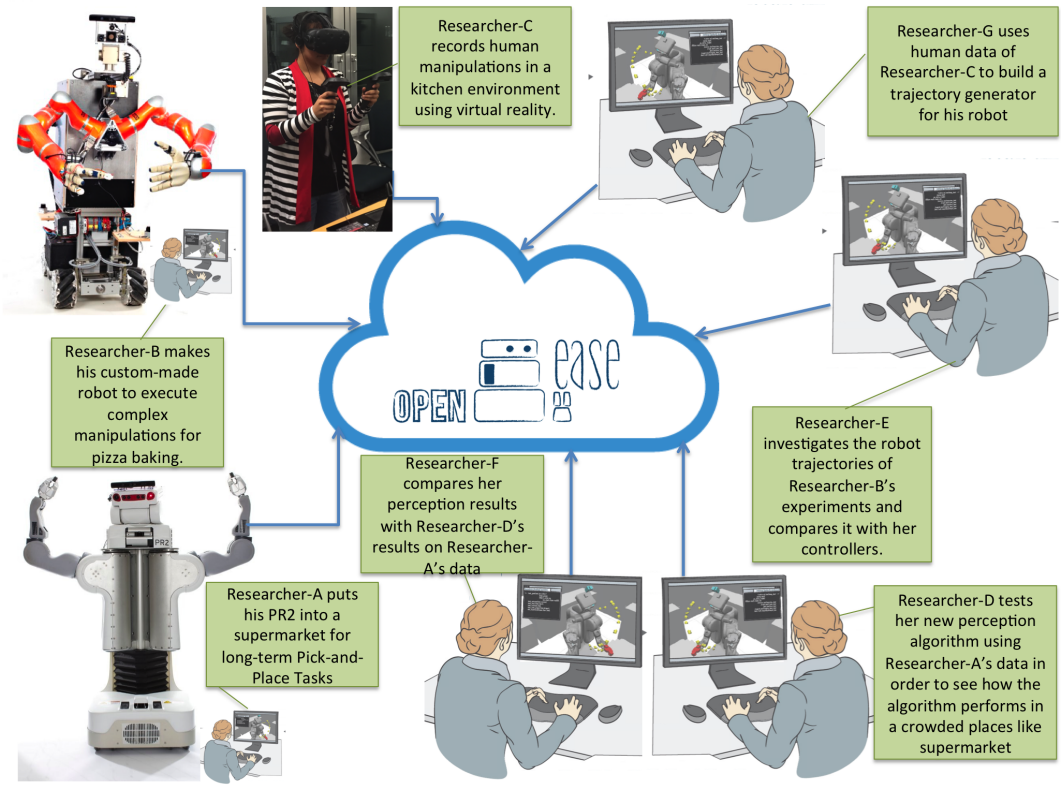


Figure 2.3: *openEASE as a platform for robotics researchers to present their research and collaborate without barriers.*

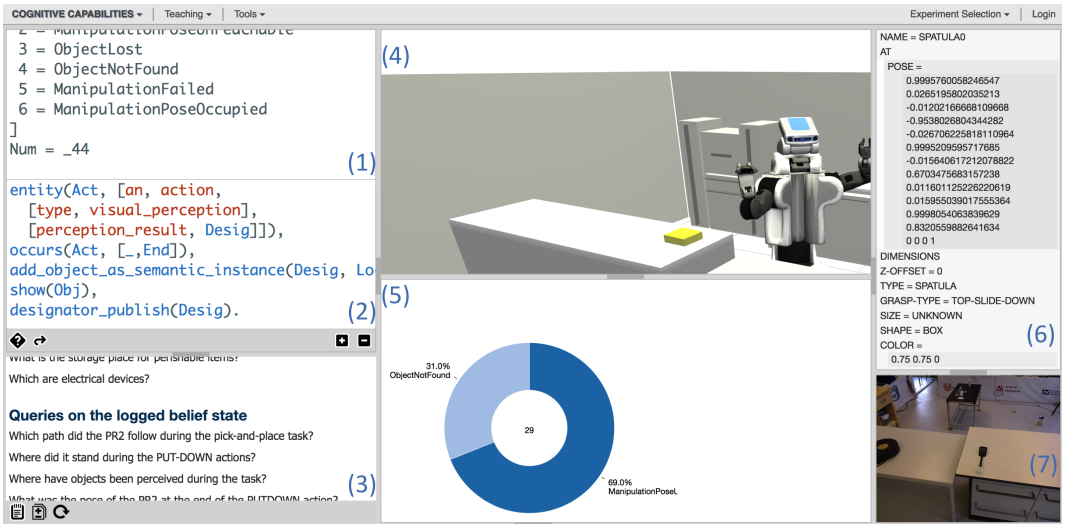


Figure 2.4: *openEASE Web Interface. (1) is the output pane of Prolog console. (2) is the user prompt of Prolog console. (3) is the pre-defined query-library for users. (4) is a 3D Canvas that depicts visual results of Prolog queries. (5) and (6) are the statistics and designators panes respectively. (7) depicts experiment images and videos taken by the agent during the experiment.*

openEASE as a Remote Knowledge Service for Robots

openEASE enables robots to use the available knowledge on the cloud and exchange knowledge between themselves for crowd-sourcing *the sense-reasoning gap* (Heintz et al., 2010). In this section, I describe how service robots benefit from such a remote knowledge source conceptually. The technical details about openEASE interface are presented in Chapter 4.1.

By having standard KnowRob ontologies about the household domain, actions, agents and environments preinstalled, openEASE already serves household robots a significant amount of knowledge for their tasks. Besides these standard ontologies, openEASE also offers ways to share user-designed ontologies between users and robots. Two different examples are depicted in Figure 2.5. Namely, *Researcher_A*'s supermarket NEEMs and ontologies are used by *Researcher_D* to test a new perception algorithm. Moreover, a household ontology designed by *Researcher_B* is used in making popcorn task, serving drink and baking pizza tasks by PR2, Toyota HSR, and UniHB Boxy respectively. On the one hand, this encyclopedic knowledge stored in ontologies covers a significant aspect of *the sense-reasoning gap*, namely symbolic knowledge that is useful for high-level planning. On the other, such knowledge does not cover all task knowledge that robots might need in their task executions.

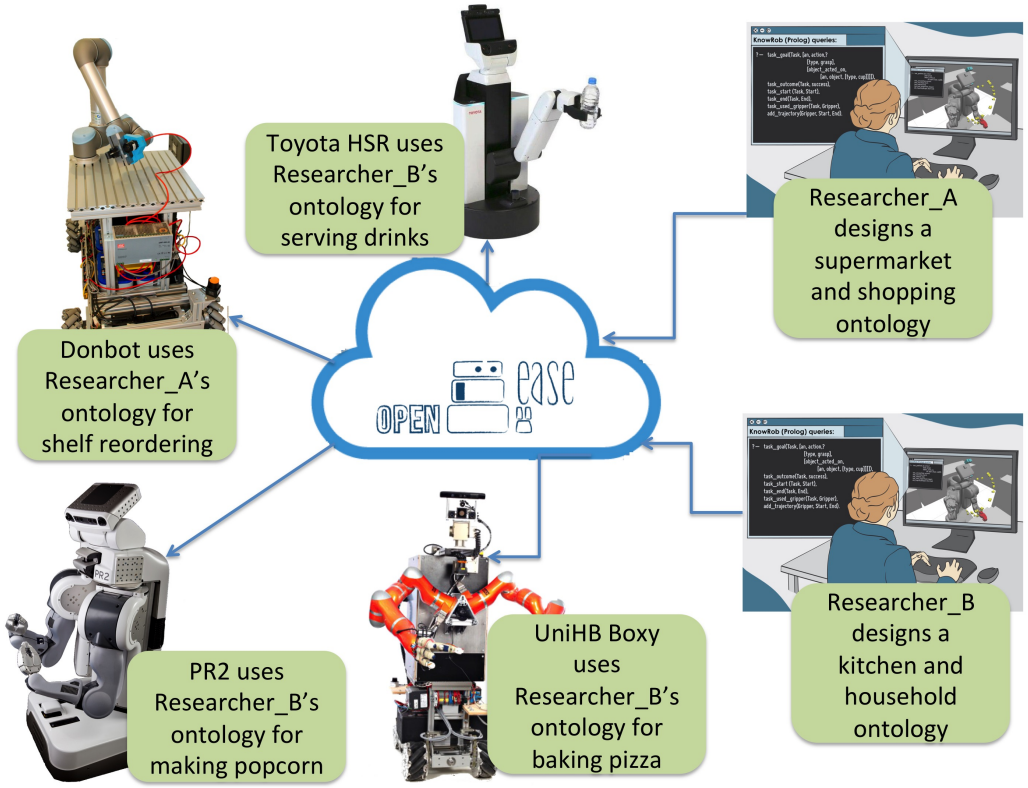


Figure 2.5: *An example illustration where Researcher A and Researcher B supply ontologies about supermarkets and kitchens respectively. These ontologies are used by different robots via openEASE.*

Symbolic and subsymbolic knowledge from similar task executions epitomizes complementary aspects of task knowledge. If a robot executing *preparing-a-breakfast* task has a set of previous execution memories from similar tasks, it can access symbolic knowledge such as *Which subtasks with which goals are executed in which order for accomplishing these tasks?*, *Where did the agent find the milk box?*, *How were the breakfast items located on the table?*, and subsymbolic data such as the base trajectory during an experiment. Robots can upload and access such experiment knowledge to/in openEASE. For this manner, openEASE uses a memory format, which links these memories to its ontological knowledge base, called *narrative-enabled episodic memory*.

2.3 Narrative-Enabled Episodic Memories (NEEMs)

As introduced in Section 1.4.2, *narrative-enabled episodic memories (NEEMs)* (Winkler et al., 2014; Beetz et al., 2018) are one of the key elements for enabling fast-learning in the presented framework. In this section, I first iterate over the requirements for a good episodic memory structure. Then, the NEEM format is presented together with example use cases such as reasoning, visualizing execution summaries, and generating statistics about executions.

2.3.1 Requirements for a Robotic Episodic Memory Architecture

Robots would profit immensely from having memories about what they did, how, why and what happened for possible improvements and development of new behaviors. Nevertheless, designing an effective memory mechanism for robots has various challenges. According to Tulving (2002), humans mostly consult to these memories whenever they start or think of executing similar actions. In other words, they retrieve the details like how they have acted, what the environment looked like using semantical cues like "*How did I cook the cake for my dad's last birthday?*". One can consider these cues as symbolic annotations used for the retrieval of details (i.e., raw data) from the episodic memory.

In NEEMs, these details index subsymbolic data such as camera images, sensory readings, and trajectories. Some characteristics of this data are as follows. 1) It is usually continuous with a fixed renewal rate (like a frequency of 50 Hz) and, thus, it

is large in terms of space in hard-drive, 2) It is raw and not readable (or analyze-able) by humans, 3) It just contains information about self and world-state and does not contain information about plans such as parameters or outcomes.

In contrast to subsymbolic data, symbolic annotations are inherently discrete since they are not sensed or calculated but rather annotated by control executives at discrete time points. For example, in a pick-and-place task, a robot possibly asserts the following annotations to its episodic memory:

- t_1: start of the navigation action to a place where the object is perceivable
- t_2: end of the navigation action to a place where the object is perceivable
- t_3: detection of the object
- t_4: successfully grasping the object
- t_5: start of the navigation action to a place where the object should be placed
- t_6: end of the navigation action to a place where the object should be placed

Because of these different characteristics, a good episodic memory design should provide different storage solutions for each.

2.3.2 A Comprehensive Episodic Memory Architecture for Robots

Modern high-level plan frameworks such as Cognitive Robot Abstract Machine (Beetz et al., 2012b) (in short CRAM) and EusLISP (Matsui and Inaba, 1990) support the implementation of complex manipulations tasks by formally defining the plans and splitting them into subactions and delegating their goals to these subactions. In particular, CRAM provides mechanisms for defining goals, implementing the reasoning processes necessary for their parameterization, and ultimately performing these parameterized tasks. High-level goals correspond to the intentions of the current execution while subactions executed to achieve these goals reflect the progress and the dynamically inferred parameterization of the task at hand. This approach allows having multiple contexts in which each subtask is executed, for example, which goals are currently active at different levels of the hierarchy.

We (Winkler et al., 2014) have implemented a module for recording NEEMs. The presented module records the symbolic data utilizing the task tree including the task

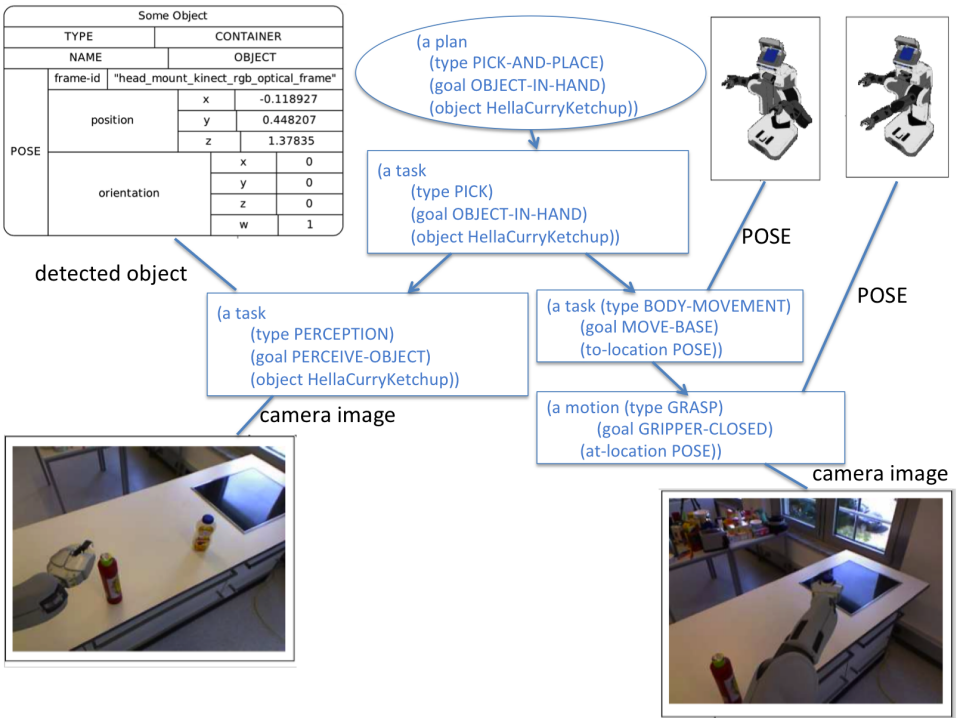


Figure 2.6: *Simplified plan event log for an* `object-in-hand` *goal achievement. A perception algorithm is employed to find the correct pose for the object in question, the robot agent navigates towards that pose, and grasps the object.*

parameterizations, failures that arose during execution, the start and end times, success states of single subgoals, and the reported progress feedback from intermediate tasks. Besides, it stores when a descriptor is created (e.g., an object's occurrence in the world is first mentioned) and when its information is updated, resulting in a change of belief about the world. This information is stored in the OWL representation language in KnowRob. Bozcuoglu et al. (2018a) have also extended EusLISP framework with a capability to record/use NEEMs to make knowledge transfer possible between CRAM and EusLISP executives.

Figure 2.6 illustrates how a `detect-and-pick` action connects hierarchically to other tasks. The task with `object-in-hand` goal is further divided into perception and grasping actions whose parameters are supplied by *designators* during run-time, together with the high-level perceptual data. The white box in the upper left visualizes the contents of the designator describing the detection of an object of type `CONTAINER`. This symbolic descriptor, like other designators, is linked to the corresponding task and stored as symbolic knowledge.

As aforementioned, the symbolic knowledge in NEEMs complements the subsymbolic data to store the complete world state according to the robot's belief as accurately as possible.

2.3.3 Recording Episodic Memories (NEEMs)

Robots that are running ROS broadcast sensor data and poses on "topics" – an asynchronous communication channel that other components (such as a logger module) can listen. Such broadcasting enables NEEMs to have virtually all pieces of information that are sent around in the robot's system. For recording topic messages, *mongodb_log* software (Niemueller et al., 2012) that stores the data in a MongoDB database is used. While this database does not support SQL for queries, it is a fast and scalable storage solution that allows recording robot data with little overhead.

In order to keep NEEMs in a manageable size, there are different options that one can opt. First, sensor data like images can be only stored for particular points in time, such as the beginning and end of a grasping action. Moreover, point are stored as depth images, which contain the same information but consume much less memory. Also, the *tf* transformations, which represent the positions and orientation of objects such as joints and links of the robot are published very frequently (at around 30-40 Hz), which makes sense for robot control during the execution, but not necessarily for reconstructing the approximate motions from NEEMs. Thus, it is possible to change the logging frequency to only store transformations which have changed more than this value with respect to the previously logged version. This strategy can help to reduce the resulting *tf* table from around 200 MB to around 30 MB for a regular pick and place task since only actual movement data is recorded.

Symbolic knowledge is asserted using prolog queries (the respective predicates are listed in Table 2.1) in KnowRob. While doing that, the roboticists must align the semantics of NEEMs with KnowRob's action ontology[2]. Some action classes that are used frequently in NEEMs are depicted in Figure 2.7. These assertions enable robots to integrate different knowledge sources for reasoning, learning, and inference. On the other hand, having two different storages would arise the question "How can these two types of data correspond to each other and used together?". A possible way to achieve that is using timepoints as a mediator between them. As an example, consider a scenario where a robot needs to retrieve its arm trajectory during a grasp action. For this, it can query for the action's time interval. Then, using the result of this query, the trajectory can be retrieved from the database that the subsymbolic data resides.

[2] `https://github.com/knowrob/knowrob/blob/master/knowrob_common/owl/actions.owl`

Predicate	Description
cram_start_action(Type, Context, StartTime, PrevAction, ParentAction, Inst)	Asserts a new action with given *Type*, *Context* and *StartTime*. There are also optional parameters *PrevAction* and *ParentAction* for maintaining task-tree hierarchy. Returns: *Inst* as the unique identifier of the new action
cram_finish_action(Inst, EndTime)	Asserts *EndTime* as the finishing time for the action *Inst*.
cram_set_subaction(Parent, Child)	Asserts *subAction* relation between *Parent* and *Child*.
cram_add_failure_to_action(Inst, Type, FailureLabel, Time, Failure)	Asserts a failure with the given *Type*, *FailureLabel* at *Time*. Returns: *Failure* which is the unique identifier of the generated failure
cram_add_triple(Inst, TripleName, Feat)	Asserts an RDF triple with *TripleName* for the action *Inst* and the action feature *Feat*.

Table 2.1: *Predicates used for recording the symbolic logs.*

2.3.4 Reasoning using Episodic Memories (NEEMs)

One can assess the completeness and comprehensiveness of NEEMs by analyzing queries can be answered based on recorded knowledge. Using predicate logic queries, durations of tasks, types, and probabilities of failures to occur, spatial reasoning to compute relations between objects and between objects and the robot's pose at different points in time, as well as the use of these inferences for diagnostic purposes can be inferred by robots. Within KnowRob framework, we (Winkler et al., 2014) have presented a set of predicates for generating such queries (listed in Table 2.2). For answering such queries, KnowRob reasoner combines knowledge from different sources such as the high-level task tree, sensory data, high-level perceptual data, and static ontological knowledge like the robot's self-model. These answers may directly be integrated into the on-going planning procedures or used for retrieving data and annotations for training machine learning models or validating such models.

As an introductory example, the average time needed to accomplish certain types of tasks can be crucial for replaning or optimization purposes. The query below returns the average time needed for the perception tasks by scanning over all the tasks with goal OBJECT-IN-HAND ?OBJ and inferring their start and end time:

```
<!-- http://knowrob.org/kb/knowrob.owl#CRAMAction -->

<owl:Class rdf:about="&knowrob;CRAMAction">
    <rdfs:subClassOf rdf:resource="&knowrob;CRAMEvent"/>
</owl:Class>


<!-- http://knowrob.org/kb/knowrob.owl#CRAMFailure -->

<owl:Class rdf:about="&knowrob;CRAMFailure">
    <rdfs:subClassOf rdf:resource="&knowrob;ActionResult"/>
</owl:Class>


<!-- http://knowrob.org/kb/knowrob.owl#CRAMPerceive -->

<owl:Class rdf:about="&knowrob;CRAMPerceive">
    <rdfs:subClassOf rdf:resource="&knowrob;CRAMAction"/>
</owl:Class>


<!-- http://knowrob.org/kb/knowrob.owl#CollisionFreeArmMovement -->

<owl:Class rdf:about="&knowrob;CollisionFreeArmMovement">
    <rdfs:subClassOf rdf:resource="&knowrob;AvoidingArmCollisions"/>
    <rdfs:subClassOf rdf:resource="http://knowrob.org/kb/pancake-making.owl#ArmMovement"/>
</owl:Class>


<!-- http://knowrob.org/kb/knowrob.owl#HeadMovement -->

<owl:Class rdf:about="&knowrob;HeadMovement">
    <rdfs:subClassOf rdf:resource="&knowrob;CRAMPerform"/>
    <rdfs:subClassOf rdf:resource="&knowrob;VoluntaryBodyMovement"/>
</owl:Class>


<!-- http://knowrob.org/kb/knowrob.owl#Navigate -->

<owl:Class rdf:about="&knowrob;Navigate">
    <rdfs:subClassOf rdf:resource="&knowrob;CRAMPerform"/>
</owl:Class>


 <!-- http://knowrob.org/kb/knowrob.owl#ArmMovement -->

<owl:Class rdf:about="&knowrob;ArmMovement">
    <rdfs:subClassOf rdf:resource="&knowrob;CRAMPerform"/>
    <rdfs:subClassOf rdf:resource="&knowrob;VoluntaryBodyMovement"/>
</owl:Class>
```

Figure 2.7: *Action classes that are used frequently in NEEMs.*

Predicate	Description
$task(Task)$	Tasks on interpretation stack
$task\_goal(Task, Goal)$	Goal of task
$task\_start(Task, ST)$	Start time of task
$task\_end(Task, ET)$	End time of task
$task\_status(Task, Status)$	Status of task (not started, ongoing or finalized)
$subtask(Task, Subtask)$	Task is a parent of Subtask
$subtask^{+}(Task, Subtask)$	Task is an ancestor of Subtask
$failure\_task(Error, Class)$	Failure of task
$failure\_class(Error, Class)$	Class of failures
$failure\_attrib(Err, Name, Val)$	Attribute of failure
$holds(occ, T_i)$	Occasions in the real world
$loc(obj, Loc)$	Location of an object
$belief\_at(event, T_i)$	Occasions in the belief state
$occurs(event, T_i)$	Events in the belief state
$object\_visible(Obj)$	Object is visible to the robot
$object\_placed\_at(Obj, loc)$	Object was placed at location
$desig\_type(Desig, Type)$	Type of plan designator
$desig\_prop(Desig, Prop, Val)$	Property values of plan designator
$robot\_pose\_at\_time(R, Fr, T, P)$	At time T, robot R had pose P in coordinate frame Fr
$obj\_pose\_by\_desig(Obj, Pose)$	Object pose from perceived designator
$lookup\_transform(F_s, F_t, T, Tr)$	Logged transform Tr from F_s to F_t at time T
$transform\_pose(P_i, F_s, F_t, T, P_o)$	Transform P_i from frame F_s to frame F_t at time T
$returned\_value(Task, Result)$	Result of task (success or fail)
$blocked\_by\_in\_cam(O, B, C, T)$	At time T, B was blocking the view of C on O
$add\_object(Obj)$	Visualizes an object in 3D
$add\_object\_with\_children(Obj)$	Also visualizes all parts of Obj
$loc\_change(Obj)$	Object changed its location
$highlight\_object(Obj)$	Highlight an object in the scene
$object\_perceived(Obj)$	Object has been perceived
$add\_trajectory(Link, St, End)$	Show trajectory of $Link$ between times St and End
$add\_diagram(Type, [DataRanges])$	Add data ranges to a diagram

Table 2.2: *Predicates for reasoning using NEEMs.*

```
?- bagof(Dur, (task_goal(Tsk, 'GOAL-PERCEIVE-OBJECT'),
               task_start(Tsk, StT),
               task_end(Tsk, EndT),
               Dur is EndT - StT), Durs),
   sumlist(Durs, Sum),
   length(Durs, Num),
   Avg is Sum / Num.

Durs = [7,7,9,7,9,7], Sum = 46, Num = 6, Avg = 7.6667.
```

Based on the result, the average is 7.7 seconds.

Moreover, it is possible to investigate which tasks has failed because of a specific failure such as 'ObjectNotFound' using the following query:

```
?- task(Task),
   failure_class(Error, knowrob:'ObjectNotFound'),
   failure_task(Error, Task).

Task = neem:'task_0487',
Error = neem:'failure_6193'.
```

openEASE can return images captured by the robot in the context of perception tasks that did not detect matching objects for the visual investigation. These images serve roboticists as diagnostic material and they can also be used by a robot to test alternative perception methods offline in machine learning applications.

2.3.5 The Connection Between Context-Specific Knowledge and NEEMs

The terms ABox and TBox are used in Computer Science to describe two different declarations in knowledge bases. A knowledge base can be conceptually represented as a combination of *terminologies* and *assertions*. TBox declarations describe *terminologies* using controlled vocabularies in terms of a set of classes and properties. ABox declarations are, on the other hand, TBox-compliant assertions in the knowledge base. For instance, a TBox is the terminology of *city* in a geography ontology. In such an ontology, possible ABoxes are city instances such as Ankara, Bremen, and Tokyo.

Similarly, the symbolic knowledge in NEEMs provides ABoxes to KnowRob ontology which defines TBoxes about household environments, agents and actions (see Section 2.2.1). Having integrated the symbolic data in NEEMs as ABoxes to the base ontology leads a uniform and integrated knowledge base for reasoning various aspects of the past experiments.

Consider a robot that is delegated a task *bringing-a-milk-box-to-the-breakfast-table*. In order not to seek the milkbox in a brute-force manner, it will load the related

NEEMs and ask where it perceived an object whose type is *CowsMilk-Product* with the following query:

```
?- owl_individual_of(M, knowrob:'CowsMilk-Product'),
   owl_individual_of(T, knowrob:'CRAMPerceive'),
   owl_has(M, knowrob:'objectActedOn', T),
   task_end(T, End),
   object_pose_at_time(M, End, Pose).
```

If this query returns a pose on a table top, the robot can try to locate the milk box again on this tabletop. On the other hand, another agent may have changed the pose of the milk box while it is not in the sight of the robot. In such a case, the robot would not find the milk box in the same location. This time, the robot would try to infer a container that is likely a storage place for milk boxes using the same ABox definition, *M*, used in the previous query:

```
?- owl_individual_of(M, MilkType),
   owl_subclass_of(T, knowrob:'StorageConstruct'),
   class_properties(T,
          knowrob:'typePrimaryFunction-StoragePlaceFor', MilkType),
   owl_individual_of(Container, T).
```

As a result, this query returns a *Refrigerator* instance based on the environmental knowledge defined in the semantic map.

Finally, robots may need to combine the knowledge in NEEMs with other knowledge sources in KnowRob. For example, in order to check whether it has necessary capabilities for *baking-pizza* previously executed by another robot, the knowledge base should include PR2's description:

```
?- owl_parse('package://knowrob_srdl/owl/PR2.owl'),
   owl_individual_of(Robot, knowrob:'PR2Robot'),
   owl_parse('/episodes/Pizza-Making/episode1/log.owl')
   owl_individual_of(Exp, knowrob:'RobotExperiment'),
   subtask(Exp, T),
   owl_individual_of(T, TaskType),
   action_feasible_on_robot(TaskType, Robot).
```

This query, first, loads the ontology that symbolically describes PR2 and a *baking-pizza* NEEM. Then, it revisits every task executed in this experiment for checking the feasibility of the execute-ability of that particular task by a PR2 robot instance.

2.3.6 Execution Summaries

Having summaries of executions as sketches can help roboticists to investigate behaviors of their robots for monitoring, debugging and further development. For

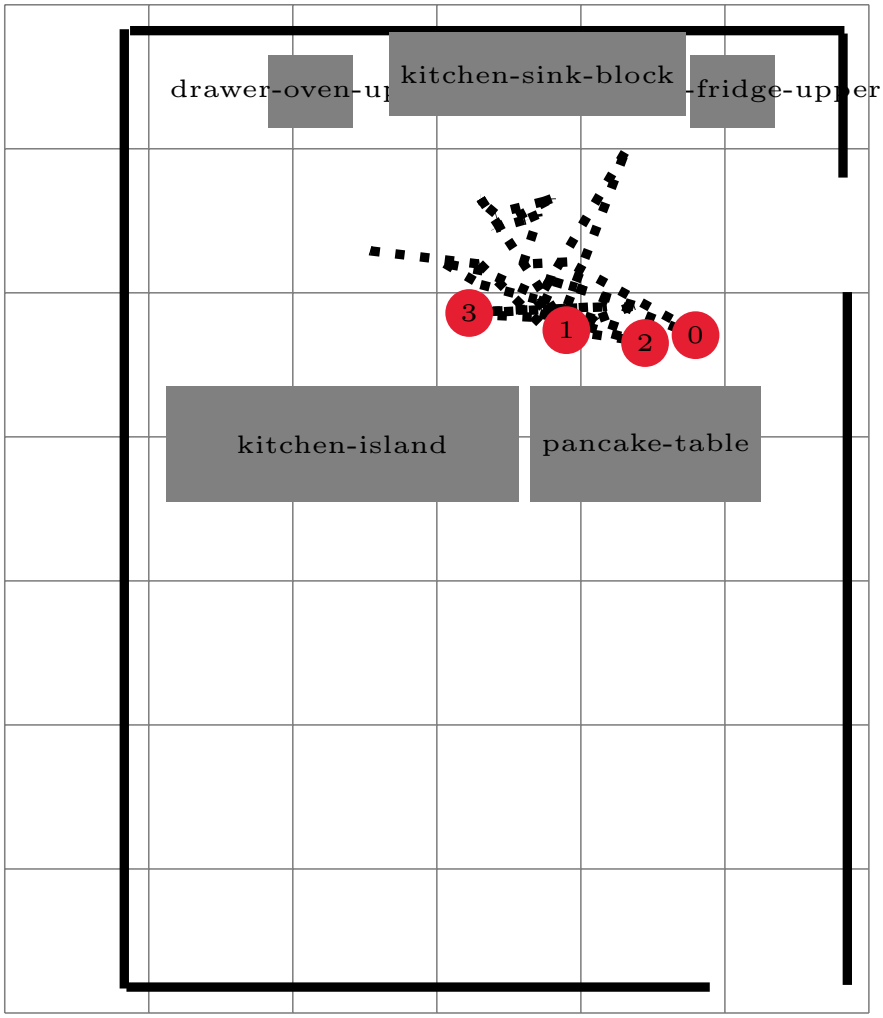


Figure 2.8: *An example 2D summary image of a pick-and-place experiment in a kitchen environment.*

this manner, I have offered a predicate *generate_jpg_summary* that generates 2-D summary images of plan executions by generating LaTeX code, compiling it and converting to JPEG format. The generated image appears at the bottom left of openEASE Web GUI after the query has been executed. Such summaries typically include the 2D map of the environment, the robot's path during the execution and the highlighted robot positions while accomplishing the task goals given as a parameter. An example summary query is:

```
?- owl_parse('pick-and-place/cram_log.owl'),
   owl_parse('knowrob://semantic_maps/room.owl'),
   generate_jpg_summary(['putdown'],
         'knowrob://summary_data/fetch-and-place.jpg').
```

which first loads the NEEM and the semantic map of the environment. Then, the last predicate call is responsible for creating a jpeg summary, in which the *PUTDOWN* positions of the robot are highlighted. The resulting image can be seen in Figure 2.8.

2.3.7 Statistical Models and Machine Learning using NEEMs

As they provide a comprehensive dataset about robotic experiments, NEEMs can be used for improving future executions of similar tasks. Bozcuoglu and Beetz (2017) provide a set of predicates for formalizing machine learning problems symbolically using predicate logic sentences formalized in KnowRob syntax. Thanks to these predicates, machine learning algorithms can be trained with features that are calculated using the subsymbolic part of NEEMs (see Chapter 6 for details). If a robot, for example, wants to grasp a cup on the kitchen counter to locate it on the table, it can train a learning algorithm by extracting pose information from the available NEEMs. For this manner, it should, first, retrieve a collection of positive executions of picking up cups. Then, for each task instance in both subcollections, KnowRob reasoner checks the robot pose relative to the object-of-interest. After that, this collection of pose information should be separated into two subcollections (positive and negative) according to whether the corresponding task is associated with any failures. Finally, any desired learning algorithm can be trained using these subcollections. These learning approaches are discussed with details in Chapter 5 and Chapter 7.

Similarly, using NEEMs, the success rate of each action type can be inferred. The probability of failure due to a given reason can be computed by the number of failed actions divided by the total number of tasks of this kind. This probability can help to model the expected behavior of the plans and to determine whether refinements are necessary. For example, the query below computes the percentage of

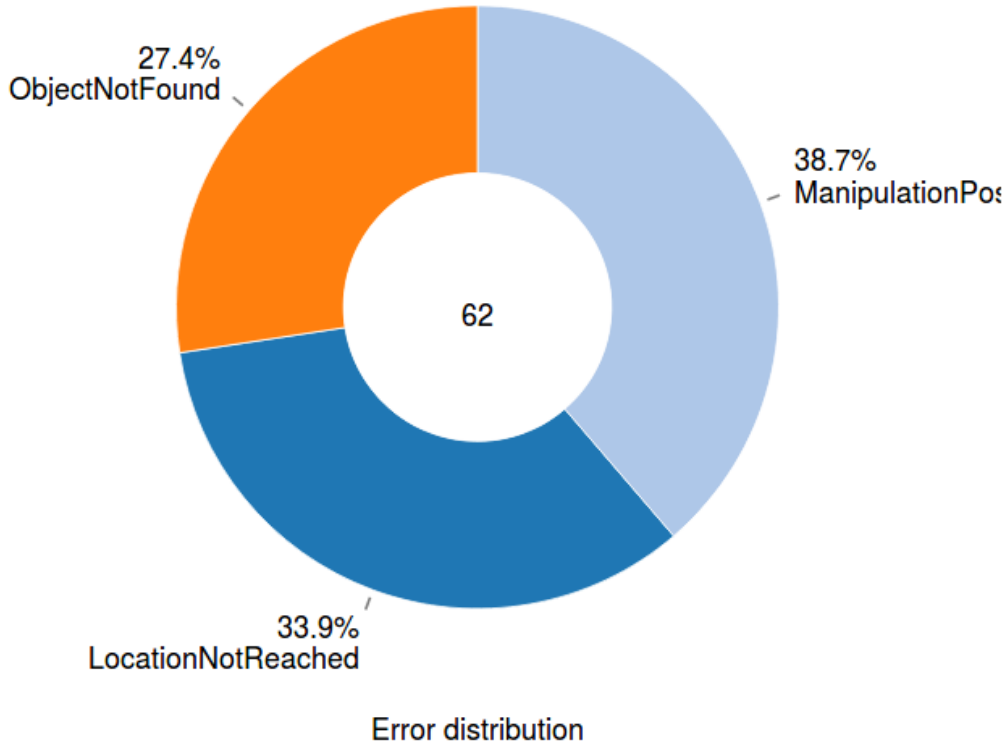


Figure 2.9: *Distribution of common failure types that occurred during task execution.*

ResolveActionDesignator tasks that failed due to an unreachable manipulation pose. Dividing the number of failed tasks (two) by the overall number of tasks of this kind (36), this leads to a probability of `Probability_of_failure = 0.0556`.

```
?- bagof(Error, (task_class(Task, knowrob:
                                  'ResolveActionDesignator'),
                 failure_class(Error, knowrob:
                                  'ManipulationPoseUnreachable'),
                 failure_task(Error, Task)), Errors),
   length(Errors, NumErr),
   bagof(Task, task_class(Task, knowrob:
                              'ResolveActionDesignator'), Tasks),
   length(Tasks, NumT),
   Probability_of_failure is NumErr/NumT.

NumErr = 2, NumT = 36, Probability_of_failure = 0.0556.
```

By using a set of NEEMs, one can also compute statistics of common error types that occur during task execution. Consider a query that reads all possible failure classes (which are subclasses of `CRAMFailure` and computes how many of these occurred:

```
?- findall(Type-Num, (owl_subclass_of(Type,'CRAMFailure'),
                      findall(F, failure_class(F,Type),Fs),
                      length(Fs,Num)), Distrib),
```

```
pairs_keys_values(Distrib,Types,Nums),
add_diagram('Failure distribution','piechart',[[Types,Nums]]).
```

The result is depicted in Figure 2.9. As shown, the `ManipulationPoseUnreachable` failure dominates, which can indicate that improving the motion planning and navigation modules promise the biggest impact on the overall performance.

Chapter three

State of the Art

This chapter overviews the literature in the domains that this dissertation contributes or makes use. The sections are structured in a similar order with Chapter 2 so that the readers can easily match the sections of both chapters and grasp the related work for the respective aspect of the presented framework.

3.1 Imitation Leaning

Imitation learning algorithms train policies based on demonstrations in order to perform similar actions. Behavioral cloning (BC) and inverse reinforcement learning (IRL) are two of the major branches in imitation learning. Using BC, an agent can learn how to act in a given state by making use of previous demonstrations. BC methods train a model to map each observation to action where observation and action pair was in the previous demonstration. They have been successfully applied in the fields such as autonomous driving (Pomerleau, 1989), solving a computer game (Ross et al., 2011) and locomotion (Kalakrishnan et al., 2009).

IRL methods, on the other hand, do not map observations to actions but fit reward functions using Markov Decision Processes (MDP) as in (Ng and Russell, 2000). Finn et al. (2016) present a work that applies IRL in dish placement tasks. Many IRL algorithms were computationally expensive because reinforcement learning has to run in an inner loop. Ho and Ermon (2016) have introduced a framework for

restoring expert policies by observing state-action pairs sampled from reinforcement learning trials.

Finally, programming by demonstration (PbD), also referred to as a branch of imitation learning, is an approach for simplifying search spaces for learning as described by Billard et al. (2008). Thomaz and Cakmak (2009) present a work in which the robot learns about object features (like what actions/effects they afford) with the help of a human demonstrator. PbD has also many applications in robotics. Pais et al. (2013) have shown how the human-robot interaction can be rewarding and interesting for humans during demonstrations.

3.1.1 Deep Imitation Learning

Although BC, IRL, and PbD are proven to work in many imitation tasks, they require retraining from scratch even a small modification in task descriptions or environmental features. In order to overcome this limitation, researchers come up with new approaches which make use of deep learning in this domain. Finn *et al.* present Model Agnostic Meta Learning (MAML) (Finn et al., 2017a) for learning a generic policy that is valid for a set of tasks. MAML finds a parameter from which the algorithm can adapt to a given task with few gradient steps. As a follow-up work, Finn et al. (2017b) demonstrate that their one-shot imitation methodology can imitate reaching, pushing, and placing tasks. Yu et al. (2018) implement a domain-adaptive version using which robots can imitate by observing human demonstration. For this manner, they have modified the inner-loss function of MAML so that temporal information is also taken into account. One of the limitations in the presented approach is learning by observing human demonstrations in third person view. Thus, people have to show demonstrations at the back of a robot.

3.1.2 Reinforcement Learning

Sutton and Barto (1998) describe reinforcement learning (RL) as a machine learning technique which maps from situations to actions for maximizing a scalar reward or reinforcement signal. There exist applications on RL methods especially in the area of motion learning. Recently, Kober and Peters (2012); Kober et al. (2013) have reviewed those methods. Among those, Ng et al. (2006) focused on learning a controller for autonomous inverted hovering of a helicopter. Michels et al.

(2005) present a system for obstacle-avoidance which makes use of both monocular vision and reinforcement learning. A sensor-based navigation framework is introduced using RL and fussy logic by Beom and Cho (1995). Moreover, there are applications of RL for different problems such as energy optimization for a walking humanoid (Kormushev et al., 2011), robot soccer (Kormushev et al., 2011), predicting depth from single monocular images (Saxena et al., 2006), and robot learning from demonstrations (Argall et al., 2009).

3.1.3 Motion Learning

Motion learning is a category of machine learning applications whose goal is to infer correct motion parameterizations depending on the conditions. Andrychowicz et al. (2017)'s *Hindsight Experience Replay* generates pushing, sliding, and pick-and-place motions by setting pseudo goals. Accomplishing these goals indicates indirectly that the real goal is also fulfilled. This method returns promising results especially in sparse environments using binary reward functions. Nair et al. (2018) introduce a Q-Filter-based approach for solving block stacking tasks by observing demonstrations. Rajeswaran et al. (2018) show how dexterous motions can be learned by adding additional terms in behavioral cloning which enables policy gradient updates for new motions.

3.1.4 The Role of Imitation for Humans

Along with many others, one can consider most of the learning tasks such as teaching, demonstrating, and guiding as social activities in which one benefits from others in terms of ability, knowledge and expertise sharing. Starting from early infancy with pure observation to more advanced ones such as course-taking and job training under supervision, humans learn from their caregivers, teachers, and friends socially (Heyes and Galef Jr, 1996).

Demonstrations are particularly important in terms of acquiring new manipulation skills. In the period of infancy, childhood, and adolescence, humans learn everyday tasks frequently using different methods. For instance, a parent can scaffold a basic stacking task with demonstrations and instructions (i.e., **semantic annotations**). In general, such scaffoldings involve showing by reason or proof, explaining or

making clear by the use of example executions. Put more simply, scaffolding can be considered as 'to demonstrate clearly to teach'.

3.2 Adaptation of Robotic Actions and Knowledge Exchange

3.2.1 High-Level Robot Plans

High-level planning is a well-studied area in the domain of intelligent robotics not just for manipulation tasks (Keller et al., 2012) but also other tasks such as navigation (Khatib, 1986; Ingrand et al., 1996) and interaction tasks (Alami et al., 2005; Thrun et al., 1999). Even at the very beginning of AI-based robotics research, roboticists have implemented programming languages such as Robot Programming Language (Park, 1983) (in short RPL) and planning frameworks such as Hierarchical Task Network (Sacerdoti, 1975) (in short HTN) according to the needs of robotics research. Today, there exist many solid planning frameworks such as Planning Domain Definition Language (PDDL) by Mcdermott et al. (1998), Structured Reactive Controllers (Beetz, 1999) (in short SRC), Cognitive Robot Abstract Machine (CRAM) by Beetz et al. (2012b), EusLisp by Matsui and Inaba (1990) and Karapinar et al. (2012)'s robust planning framework for service robotics applications.

Using these, roboticists have shown many applications such as using elevators and fetching a sandwich from a restaurant (Saito et al., 2011), making pancakes (Beetz et al., 2011a), cutting vegetables (Ogura et al., 2006) and serving a drink (Waibel et al., 2011).

3.2.2 Affordance Modeling

The term *affordance* is used to define the relationships between a robot and its environment in terms of effects that the robot can generate. Recently, Zech et al. (2017) have published a survey on affordance-based computational models with recent studies, a systematic classification of them, and on-going challenges. One of the most inspirational studies on the field is by Arbib et al. (2000) in which they study how the human brain handles behavior learning, particularly grasping, using imitations. Ramirez-Amaro et al. (2017) develop semantic models of human activities in the form of affordances in order to transfer these skills to robots,

Levine and DeJong (2006) demonstrate an explanation-based learning methodology which attempts to learn quantitative relationships between the actions an agent performs and the state variables describing the world. Bierbaum et al. (2009) extracts grasp affordances for unknown objects using tactile exploration. Similarly, Kuniyoshi et al. (2005) propose an object affordance investigator for a multi-fingered robot hand using haptic features in an approach based on high-order local autocorrelation, principal components analysis, and mean-shift clustering.

Moreover, there are studies on Gaussian Mixture Models (similar to the methodology described in Chapter 7) such as learning by demonstration (Chernova and Veloso, 2007; Khansari-Zadeh and Billard, 2011; Calinon et al., 2007) and model-based control (Nguyen-Tuong et al., 2009). The ARPlaces approach by Stulp et al. (2012) models probabilities of task success before execution based on parameters such as relative locations of the robot and the objects it needs to grasp. This study is particularly related to this dissertation since they use subsymbolic action parameters and action effects, rather than STRIPS-style abstract features (Fikes and Nilsson, 1971). The abstractions in STRIPS often fail to capture small environmental properties to help to boost performance such as the existence of a place from which several objects can be grabbed from which obviates the need for several navigation actions (Stulp et al., 2012). Moreover, probabilistic representations of locations, in terms of their effects on actions, is deployed in a relatively large knowledge base to parameterize vague actions (Beetz et al., 2012a).

3.2.3 Adaptation of Robotic Actions

Reusing the data inside execution logs and adapting them to new circumstances has been investigated in various studies. Use cases such as opening the same furniture by a different robot, have been demonstrated in the scope of RoboEarth project (Waibel et al., 2011; Tenorth et al., 2013). On the other hand, these use cases do not include analyzing differences between the physical properties of the environment and adapting the data accordingly as I present in this dissertation.

In his dissertation, Huckaby (2014) presents a framework called GTax in order to generalize manipulation skills so that it is easy to modify these skills according to needs. Bocsi et al. (2013) showcase a method that improves the learning performance by using additional experiment data from other tasks performed by the same robot or even by a different robot.

3.2.4 Knowledge Sharing

Knowledge sharing between agents has been a popular topic since the late 90s in robotics. Wakita et al. (1998) proposed a methodology by enabling information sharing between agents for ensuring human safety while performing service tasks.

For the knowledge exchange protocol, Huckaby and Christensen (2013) present a Systems Model Language which is used for defining capabilities in terms of skills and action primitives. Using this system, robots can figure out if they have certain capabilities by analyzing if they can execute the action primitives. Vuga et al. (2015) describe a methodology to generate semantic probability models of human-executed actions. These models can be used by robots to execute similar actions. The functional object-oriented network (FOON), a knowledge representation framework, is introduced in (Paulius et al., 2016). Using FOON, robots can decompose a given task's goals, find the object-of-interest, and adapt the goal according to the current state for generating proper manipulation motions. Lastly, Pangercic et al. (2012) and Rusu et al. (2009) provide methodologies to infer semantic annotations of environments via 3D perception systems and to integrate these annotations to symbolic-level knowledge bases tightly.

3.2.5 Adaptation and Knowledge Exchange in Humans

Household tasks are highly challenging in terms of manipulation skills and dexterity. To perform these tasks under varying conditions and environmental properties, humans use advanced cognitive abilities such as reasoning, recalling and learning to adjust themselves to new conditions and environmental properties (Anderson et al., 2000). Moreover, humans can acquire and generalize knowledge using only very few examples (Anderson and Schooler, 1991), and also across different modalities such as watching videos (Anderson et al., 2004b), learning from instructions (McLaughlin, 1965). They use memories on their activities heavily for improving them or adapting on different conditions, and, also, they abstract and consolidate knowledge through dreaming (Tulving, 2002). Additionally, the human brain can generate a persuasive presence and use of common sense and naive physics knowledge (Hegarty, 2004). For instance, a person already has a detailed intended arrangement of utensils on the table when he is asked for setting a table.

3.3 Prospection

A robotic prospection is, in a way, a fast simulation mechanism using which robots test their action parameters and estimate their effects Kennedy et al. (2008) propose a framework using which robots can predict how humans will act in human-robot teams by performing a simulation, given a cognitive model of the human. In (Cassimatis et al., 2004), an architecture is proposed in order to integrate reasoning, planning, sensing and motion planning algorithms by composing them from strategies for managing mental simulations. Roy et al. (2004) present a mental imagery system for an interactive robot to maintain a "mental model" of its physical environment by coupling the perception. Using this, the environments can be seen from different perspectives, providing the basis for enrichened language comprehension and production. An autonomous learning model for improving the manipulation ability of an iCub using mental imagery is described by Di Nuovo et al. (2013).

3.3.1 Prospection in Humans

The ability of prospection, which highly relies on past experiences and observations, is a skill to imagine taking specific action and predict its consequences (Taylor et al., 1998). These predictions help humans' to correctly parametrize and finetune their actions before executions.

From a pragmatic point of view, one can consider the ability of prospection as substitutes for real-world experiences. As stated by Kappes and Morewedge (2016), humans choose most of the time mentally simulating actions instead of presently engaging, by means of distraction, coping, or preparation for the future. Consequently, many psychologists such as Bass et al. (1972); Kappes and Morewedge (2016); Hamilton et al. (2014) argue that the prospection evokes similar cognitive, physiological, behavior consequences as executing the corresponding action in reality. In a way, this means that such simulations generate short-term memory entries using which humans can infer what they did, why, how, what happened, and what they saw as they do with their memories from reality.

3.4 Knowledge Representation and Processing

3.4.1 Task Knowledge for Robots

Acquiring knowledge and reasoning are two of the key capabilities of cognition-enabled robot control. Stanford's Shakey (Nilsson, 1984) can be named as a pioneering work where the map of the block world that Shakey operates in is supplied as first-order logic sentences. There are also studies by Bekey et al. (1993) where a knowledge-driven grasp planner is presented and by Hoffmann and Pfister (1997) where a fuzzy knowledge base for mobile robots is proposed.

As robots start to accomplish more and more complex tasks, deeper knowledge is needed to perform these task executions in different circumstances. On one hand, supplying such knowledge from various sources is an obvious challenge for roboticists (Bateman et al., 2017). On the other, yet a greater challenge is to implement control programs that are able to harmonize and make use of such heterogeneous knowledge bases as in (Nyga et al., 2017).

Towards overcoming these challenges, roboticists develop open-source frameworks such as KnowRob (Tenorth and Beetz, 2013), AfRob (Varadarajan and Vincze, 2012), and the Upper Ontology/Methodology (Schlenoff et al., 2012) which provide mechanisms and interfaces for equipping robots with symbolic knowledge. This knowledge can be reasoned by robotic agents using first-order logic queries. An example would be "What is the likely location of a milk box?" which can be formalized with an unbound variable *Loc* in Prolog syntax as follows:

```
?- type_of(milkbox_Abc, Typ), likely_loc(Typ, Loc).
```

wherein the first predicate, the type of *milkbox_Abc* (*Typ*) is bounded as *perishable_item*. Then, a *likely_loc* relation is searched through between the type *perishable_item* and the unbound variable *Loc*. KnowRob reasoner returns *Fridge* as the answer.

Finally, Lemaignan et al. (2006) present an upper ontology called MASON for data formalization and sharing in manufacturing applications. They have also shown that this ontology is suitable for automatic cost estimation and semantic-aware multiagent system for manufacturing.

3.4.2 Cloud Robotics

Household tasks are usually underspecified. Thus, context-specific knowledge and learning capabilities are essential in order to understand their essence and perform them successfully. In general, robots' own computing units are limited for carrying out such computationally expensive processes because of space, energy, and thermal restrictions. In order to overcome them, Kuffner (2010) coined the idea of cloud-enabled robots which use remote computing facilities and services in order to offload their high-load processes.

Cloud robotics aims for offering remote services such as a knowledge base and reasoning engine (Beetz et al., 2015), natural language processing (Misra et al., 2014; Sung et al., 2014) and concept learning (Saxena et al., 2014). Recently, Saha and Dasgupta (2018) surveyed the latest advancements in the field of cloud robotics. This survey contains significant studies such as RoboEarth cloud engine (Riazuelo et al., 2015), which is a framework that stores data injected by humans and robotic agents in a machine-readable format. This stored data includes software components, navigation maps, action recipes, and object models. Rapyuta (Mohanarajah et al., 2014) is a follow-up work of RoboEarth in which robots offload heavy computation by providing secured customizable computing environments in the cloud. Lastly, a cloud-based grasping scheme which uses Google Services is proposed by Kehoe et al. (2013).

Sung et al. (2018) propose a crowdsourcing web platform called *Robobarista*. Although the idea is very similar to the proposed imitation layer, *Robobarista* enables crowdsourcing using manual labeling and natural language processing instead of actual demonstrations inside a photorealistic virtual-reality environment. Recently, Mandlekar et al. (2018) have also presented a crowdsourcing platform, called *RoboTurk*, for learning applications. This platform enables users to gather training data by teleoperation instead of demonstrations in virtual-reality.

3.5 Episodic Memory Architectures

There exist different episodic memory architectures proposed in the domain of cognitive architectures. One of the earlier ones that stimulates humans' working memory is presented by Laird et al. (1987). It contains procedural and declarative knowledge along with episodic memory. Namely, it contextualizes a *context stack*,

which specifies active goals, problem spaces, states and operators of the embodied agent, *objects*, which are denoted with attributes called *values*, and preferences, which give the procedural search-control knowledge.

ACT-R (Anderson et al., 2004a) is another framework that has the notion of episodic memory. In contrast to Laird et al. (1987), it has separate memory for declarative and procedural knowledge, which contain just facts and individuals (or things). It was used in different cognitive applications such as choosing an association of a concept (Anderson and Reder, 1999) and simulating the theory of *serial memory* in psychology (Anderson and Matessa, 1997).

ICARUS (Langley et al., 2009), an integrated cognitive architecture for agents, has two memory structures. The first one is the conceptual memory which stores knowledge about features of things and their relationships. Secondly, it has a skill memory that stores knowledge about achieving goals. Each of these memories has both a long-term and short-term submemory.

3.5.1 Episodic Memories for Robots

In the domain of robotics, a memory system should take into the account of properties of physical robotic agents, scalability issues due to storage and processing constraints. An early study in this field was done by Beetz (2000) using a simulated robot in a simple environment for navigation tasks.

For subsymbolic data logging, Niemueller et al. (2012) have presented a robust logging system for low-level sensor signals into a NoSQL database. Hilbert and Redmiles (2000) describe the benefits of symbolic logging and subtask transformation into streams of interest by selecting and storing them according to the current needs. They use this approach in order to summarize action sequences as tasks and to distinguish these sequences based on probability.

Finally, Rothfuss et al. (2018) present an architecture called Deep Episodic Memory in order to represent robotic experiences in a human-like memory which enables efficient encoding, recalling, and predicting action executions. Using this architecture, the authors show that it is possible to infer similar experiences, to reconstruct episodes in a certain extent, and, even, to predict future.

3.5.2 Episodic Memory in Cognitive Science

Tulving et al. (1972) have coined the term *episodic memory* by making distinction between *semantic memory* and *episodic memory* 46 years ago. Episodic memory is used to consciously recall previous experiences from memory (e.g., recalling a recent business trip to Tokyo), whereas semantic memory is the ability to store more general knowledge about things, facts, and actions (e.g., the fact that Skytree is in Tokyo). Although the episodic memories are limited to only actions and materials (or objects) associated with actions in his original work (Tulving et al., 1972), he refined and elaborated his theory for episodic memories over the years. In the early 2000s, he explains his current view on episodic memories with the following paragraph:

> Episodic memory is a recently evolved, late-developing, and early - deteriorating past-oriented memory system, more vulnerable than other memory systems to neuronal dysfunction, and probably unique to humans. It makes possible mental time travel through subjective time, from the present to the past, thus allowing one to re-experience, through autonoetic awareness, one's own previous experiences. Its operations require, but go beyond, the semantic memory system. Retrieving information from episodic memory (remembering or conscious recollection) is contingent on the establishment of a special mental set, dubbed episodic "retrieval mode". Episodic memory is subserved by a widely distributed network of cortical and subcortical brain regions that overlaps with but also extends beyond the networks subserving other memory systems. The essence of episodic memory lies in the conjunction of three concepts:self, autonoetic awareness, and subjectively sensed time *(Tulving, 2002)*.

Probably, the most related aspect for robotics is facilitating agents with mental time travel to the past and re-experiencing previous actions in "retrieval mode". Such an ability can lead or assist many cognition skills in robotics as well. For instance, humans can reason on what was wrong in this particular execution and can infer what could have prevented this by living through past failures of a certain task. Mental simulation and reasoning capabilities built upon a comprehensive episodic memory architecture can also enable robots to live through their past executions and reason on what went wrong and how to resolve or improve that.

Chapter four

Cloud Interface

openEASE, as the cloud robotics platform, is used in this dissertation to store *narrative-enabled episodic memories* (NEEMs) and exchange them between agents, and, also, offload computationally-expensive mental simulations. openEASE as a platform is introduced in Section 2.2.1.

In this chapter, I describe how robot control executives can interface with this platform and record and access experiences in the format of NEEM.

4.1 Communication with openEASE

In Section 2.3.2, I have described how NEEMs can be useful within the robot control loop based on ROS communication layer. Although using ROS eases the integration of this logging module with robot software, it still requires a substantial effort concerning the factors such as mapping inner data structures of the robot with the respective concepts in the symbolic knowledge base and establishing an on-site knowledge base to keep the saved episodic memories.

To minimize these efforts, Bozcuoglu et al. (2018a) present an interface to openEASE cloud engine which makes possible to record episodic memories directly on the cloud. Such an interface is not only ideal for lowering integration efforts but also for enabling robots to exchange their episodic memories.

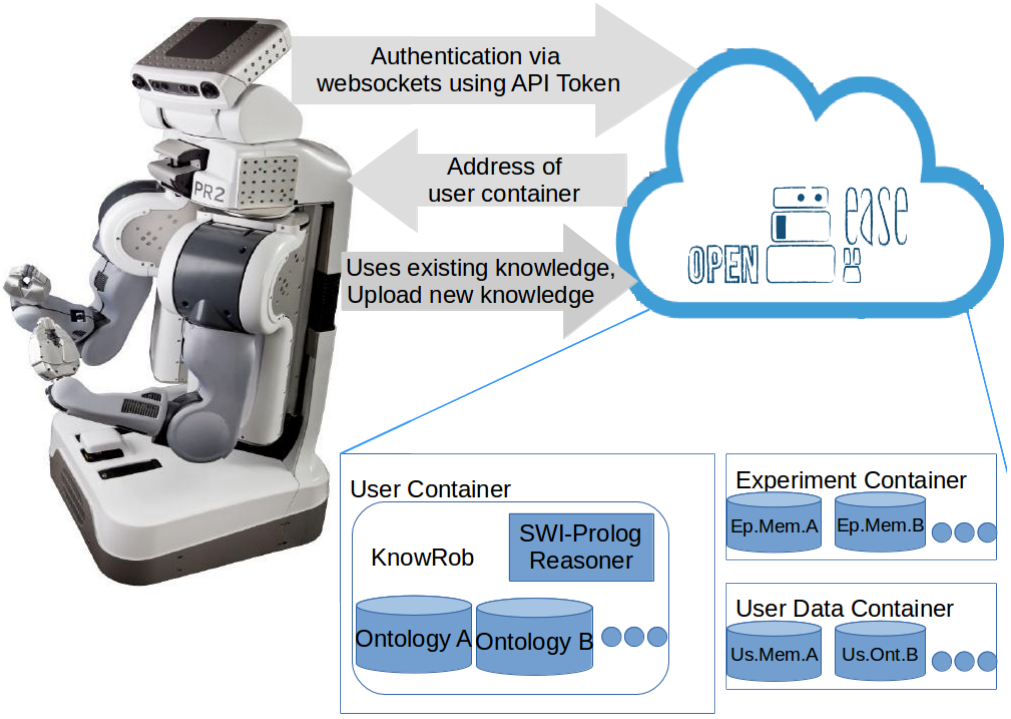


Figure 4.1: *Communication with openEASE using the robot interface.*

openEASE uses Docker framework (Merkel, 2014) for creating virtual storage and processing units. As depicted in Figure 4.1, robots initiate communication by authenticating themselves using their unique API tokens (Horstmann, 2015). These tokens are generated using the web interface. openEASE returns a web socket address as the response in the case of successful authentication. By using this address, robots can interface with their containers and system-wide containers. The user-specific containers have the KnowRob executive and standard ontologies preinstalled. Additionally, they have read access to a system-wide container that has standard execution logs together with read/write access to user-specific data containers where user-specific ontologies and execution logs reside.

Robotic applications can interface with openEASE using a C++ API[1] which offers an abstraction layer to the connection protocol mentioned above. Besides, there exists a set of SWI-Prolog predicates[2] which offers a similar abstraction layer inside KnowRob querying infrastructure.

[1] `https://github.com/asilx/openease_robot_interface`
[2] `https://github.com/knowrob/knowrob_addons/tree/master/knowrob_cloud_logger`

For reasoning and learning using existing NEEMs in openEASE, there are no additional requirements other than the formalization of the queries and the use of standard KnowRob predicates. However, for accepting new execution logs from users, it is expected to comply with the standard action ontology`https://github.com/knowrob/knowrob/blob/master/knowrob_common/owl/knowrob.owl`.

4.2 Recording Episodic Memories (NEEMs) to the Cloud

In Section 2.3.3, I have outlined how robots can log NEEMs to KnowRob and local mongoDB instances. Alternatively, they can also log NEEMs directly to their data container in openEASE via the robot interface using assertion queries. Subsymbolic data is stored directly in the mongoDB container of openEASE using the dedicated port (Figure 4.2). Both symbolic and subsymbolic data are labeled with timestamps in client machines. Thus, roboticists are expected to synchronize the clocks of their respective symbolic and subsymbolic logger nodes. Within ROS, this synchronization is relatively straight forward using ROS clock server[3].

At runtime, the robot control executives record such memories by using Prolog predicates described in Table 2.1 to their data containers where they have read and write access. Later on, they can load these NEEMs to their KnowRob executives. According to the demand, openEASE administrators may also make these non-standard NEEMs public so that other accounts have read access.

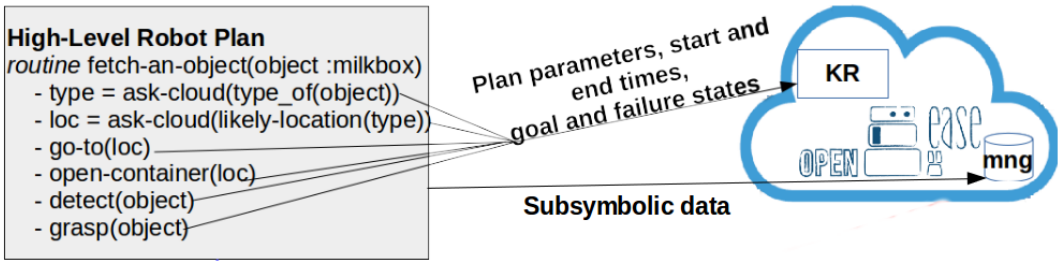


Figure 4.2: *Logging plan executions to openEASE (KR and mng are abbreviations for KnowRob and mongoDB respectively).*

[3] http://wiki.ros.org/Clock

4.3 Using NEEMs and Ontologies Employed in openEASE

As NEEMs available, robots have an extra source of information besides encyclopedic and environmental knowledge. Using these "memories," robots can reason about which parameters led to successful executions and which conditions resulted in failures.

For instance, a service robot can ask how it grasped an object of a particular type successfully in previous executions. In Figure 8.2, PR2 asks how it grasped *pancake mix* with success. As a result, openEASE returns a pose of the robot together with the action parameters.

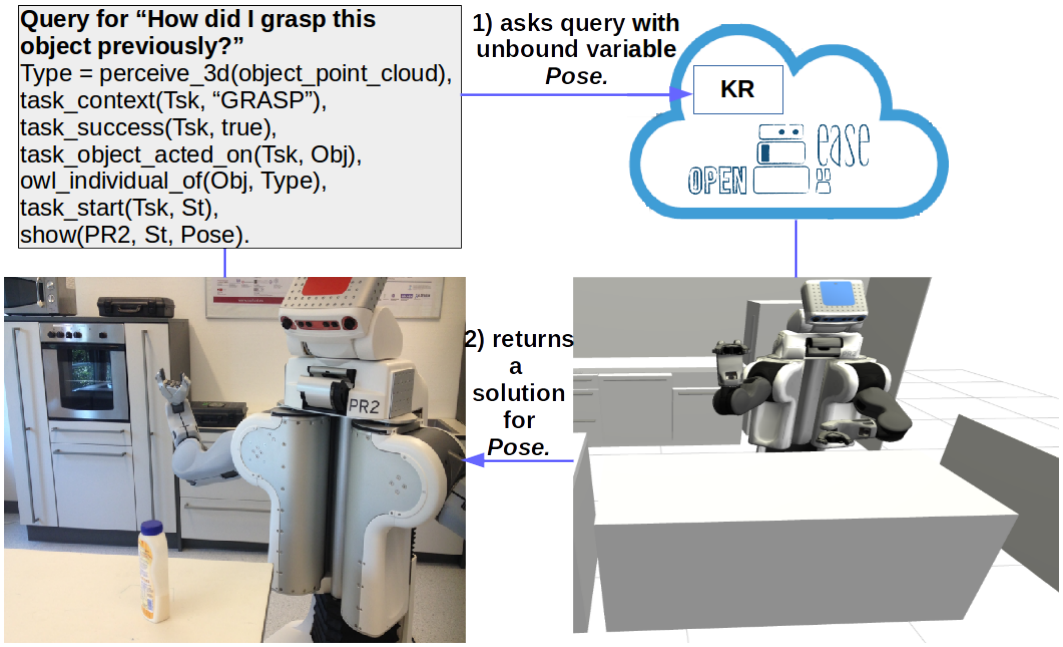


Figure 4.3: *An example use case where PR2 asks its pose which it could grasp the object successfully to openEASE equipped with NEEMs.*

4.4 A Field Robotics Use Case

Accidents and disasters during events such as skiing or avalanches hitting humans in mountainous areas cause casualties in human lives and properties. For such mission-critical scenarios, it is challenging to dedicate human rescue teams in every

risk areas due to logistical and financial difficulties. Recently, search-and-rescue (SR) professionals and governments show a keen interest in teleoperated robotic systems which let them carry on these tasks more efficiently and remotely from risk-free zones. On the other hand, during the teleoperation technical issues like communication delays due to low-bandwidth or incomplete network coverage can still hurt the outcome substantially.

During SR missions, professionals tend to lead a systematic search, e.g., by foot or by specialized vehicles and with rescue dogs, that may often take time and cause inefficiencies and delays in the course of operations especially if it is in more massive terrains. A more promising approach would be robotic systems working alongside and assisting humans with their complementary capabilities, e.g.autonomously performing tasks, collecting data quickly, communication in low-bandwidth and abstracting information pieces, highly required in rescue missions.

A European FP7 Research Project, called "Smart collaboration between humans and ground-aerial robots for improving rescuing activities" (SHERPA) (Marconi et al., 2012), aims for offering such a multi-robot system where robots can act semi-autonomously using leading-edge reasoning and planning techniques. SHERPA robots are equipped with different hardware modules to accomplish specific tasks in the context of the mission, e.g., a camera for taking pictures or a receiver for receiving signals from avalanche beacons. There exist different types of aerial and ground robots that are specialized to carry different tasks (Figure 4.4).

In (Yazdani et al., 2018, 2019), I have extended openEASE for keeping a central belief state accessible and modifiable by the SHERPA team (including the human operator). During the mission, robots additionally record subsymbolic motoric and sensory data logs annotated with symbolic annotations as NEEMs in openEASE. The human operator can access the shared belief state and the “episodic memories” by using a dedicated web interface From this interface the she can query these “memories” regarding what the team did, why, how, what happened, and what robots sensed. It is also possible to click-and-command robots with openEASE by analyzing 3D rendering of the terrain together with robot sensory data. The robotic team members can query the central belief state and NEEMs via a dedicated Prolog service. Thus, SHERPA robots are able to issue queries for answering questions or making decisions during task execution. An example of such is “How has a drone called *blue wasp* scanned the area of *FrozenLake_rFdy*?” shown below:

```
?- owl_individual_of(Act, knowrob: 'ScanningArea'),
   rdf_has(Act, knowrob:'objectActedOn', sherpa:'FrozenLake_rFdy'),
   rdf_has(Act, knowrob:'performedBy', sherpa:'Blue_Wasp'),
   task_start(Act, Begin),
   task_end(Act, End),
```

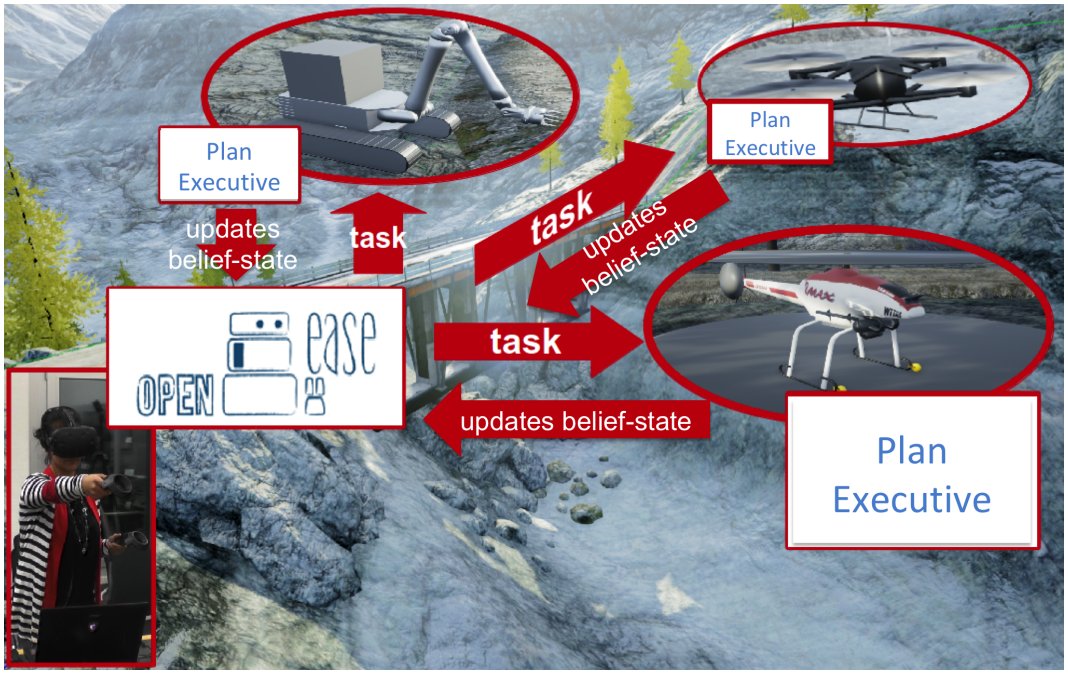


Figure 4.4: *The Proposed SHERPA Human-Robot Mixed Search-and-Rescue Team coordinating an operation within a virtual reality-based simulation.*

```
show(trajectory('Blue_Wasp''), interval(Begin,End, dt(0.5))).
```

This query returns and visualizes a scanning trajectory which can either be used by the human to monitor robot actions (see Figure 4.5 for a trajectory visualization) or by another robot to change or adjust for better results depending on time requirements or efficiency of the scanning process.

Figure 4.5: *A query for visualizing the trajectory of blue wasp while flying over GISPolygon1*

Part II

Learning Manipulations from Demonstrations Recorded in Virtual Environments

Chapter five

Fast Imitation Learning

In this chapter, I present how imitation enables robots to imitate human actions executed in virtual reality (VR) via meta-learning. I validate this approach by presenting experiments in which Toyota HSR imitates novel tasks recorded in VR *narrative-enabled episodic memories* (NEEMs).

The research and scientific results described in this chapter is also presented in (Bozcuoglu et al., 2019b).

5.1 The Importance of Fast Action Learning in Service Robotics

In order to have robots serving in everyday life, they should be capable of improving their manipulation skills according to the needs of their dependents. A possible approach to having this capability is to establish "skill markets" where users can download the manipulation add-ons to their robots as they download apps for their smartphones. On the other hand, this would not completely solve the problem of how the robots master these manipulations because every home has a unique setting of furniture and objects. Consider a case that a senior wants her brand new service robot to serve coffee. Even she could download an add-on for that, every cup and every kitchen have variances which prevent a standard add-on to work for every cup in every home. Moreover, we, as roboticists, cannot expect users to program their robots according to their habitat or collect data for machine learning because these

activities require a certain degree of technical expertise.

Thus, I believe that the best strategy to solve this problem is establishing imitation learning mechanisms that are

- **easy-to-demonstrate**: Users cannot demonstrate or teach the actions as a person who knows about programming and robot kinematics would do. Thus, the mechanism itself should identify the essence of new actions without a good set of demonstrations.
- **fast**: One cannot expect ordinary people to gather a large set of demonstrations for an "easy" task such as serving a coffee.

5.2 Imitation using Virtual Reality (VR) Demonstrations

There are two ways of gathering human demonstrations for learning applications. The first one is making the robot to observe the demonstrator using its perception system. As the perception in unstructured environments is an open-research question by itself, one cannot expect a demonstration set without noise or false perceptions. Such challenges with data usually increase the size of demonstration sets or need dedicated mechanisms to identify error-less perception conditions in the given environment.

The second way is using a tracking system with markers or multiple-cameras. With this way, a higher quality of demonstration set can be gathered without errors due to perception or action recognition. On the other hand, having such systems, which are relatively expensive, along with the robotic hardware would hurt the budget of consumers and prevent the service robots from becoming mainstream products for a long time.

Alternatively, virtual-reality (VR) devices become mainstream consumer products (Figure 5.1) in such a way that people start to buy VR headsets for their homes and to make social gaming events in commercial virtual reality rooms.

As virtual reality games can provide *photorealistic* and *physics-enabled* environments where motions and events can be detected and tracked, it is possible to develop games with purposes for data collection where players can demonstrate tasks in a virtual environment. The VR NEEMs would also be nearly error-free since the virtual environments are controlled and the developers have access the ground truth via game engine. Moreover, VR demonstrations do not require the robot and the demon-

Figure 5.1: *The crowd tries Samsung's latest smartphone which can be used as a VR Headset (The photograph is a courtesy of Maurizio Pesce).*

strator to share the same environment. This feature enables remote demonstrations by other people for the people who are physically-limited or handicapped.

5.3 Episodic Memories from Virtual Reality

NEEM loggers have access to different knowledge sources during a robotic experiment or a VR demonstration. During a robot experiment, they can interface with the processes of the control executives and access to insights such as motion parameters, what the robot did, why, how, what happened, and what it saw (Figure 5.2a). On the other hand, they can not access the world state or the physical events that take place, such as changes in the state of a kitchen door.

VR demonstrations take place in environments that are rendered and orchestrated by a game engine. Having a controlled environment means that the logger can interface with this engine to reach insights about the physical events and the state changes in environments (Figure 5.2). On the other hand, the logger cannot interface with

the demonstrator's brain for accessing insights about her task planning or motion parameters in contrast to robot experiments.

In short, the loggers have access to "brain" during robotic experiments and to "physical world" during VR demonstrations. Thus, there exist content-wise differences between NEEMs from these occasions as presented by us, Haidu et al. (2018). In particular, the symbolic part in robot NEEMs contains high-level plan information of robot executions such as task hierarchy from high-level plans, plan parameters, failure and success states of each goal. On the other hand, the symbolic part in VR NEEMs contains force-dynamic event information and furniture state changes coming from the game engine.

The subsymbolic data in robot NEEMs contain joint states, trajectories, sensory data and camera images whereas the viewpoint of the demonstrator, marker, and palm poses are stored inside VR NEEMs. Similar to the robot ones, both symbolic and subsymbolic parts are tagged with timestamps so that these two sources can be linked together for answering queries like *"How did the state of the drawer change while the demonstrator was touching its handle?"*

NEEMs are crucial to the crowdsourcing aspect of the presented methodology. Consider a scenario that you have realized that your new learning scheme would perform better if you supply a new feature, say camera pose. If you had a case-specific logger in your game system, you would ask the public to update their games and record new data. Instead, people can submit their NEEMs with a particular sequence of actions only once in our approach. With the uploaded NEEMs, roboticists can flexibly change the learning features and, even, use different action types.

5.4 One-Shot Domain Adaptive Meta-Learning from VR Demonstrations

Robots cannot just re-execute human demonstrations as they are since the state space and action space of service robots and humans are substantially different. Domain-adaptive techniques approach this problem by learning mappings from the human domain to robot domains.

```
<knowrob:Reaching rdf:about="&knowrob;Reaching_NLUFSXJD">
  <knowrob:bodyPartsUsed rdf:resource="&pr2;pr2_right_gripper"/>
  <knowrob:endTime rdf:resource="&knowrob;timepoint_1521642317"/>
  <knowrob:goalLocation rdf:resource="&knowrob;Pose_KXTNJZUS"/>
  <knowrob:objectActedOn rdf:resource="http://knowrob.org/kb/iai-kitchen.owl#EdekaRedBowl_NSVUDYQT"/>
  <knowrob:startTime rdf:resource="&knowrob;timepoint_1521642298"/>
  <knowrob:taskContext rdf:datatype="&xsd;string">DummyContext</knowrob:taskContext>
  <knowrob:taskSuccess rdf:datatype="&xsd;boolean">true</knowrob:taskSuccess>
  <rdf:type rdf:resource="&owl;NamedIndividual"/>
</knowrob:Reaching>

<knowrob:LiftingAGripper rdf:about="&knowrob;LiftingAGripper_LMQSNKGO">
  <knowrob:bodyPartsUsed rdf:resource="&pr2;pr2_right_gripper"/>
  <knowrob:endTime rdf:resource="&knowrob;timepoint_1521642525"/>
  <knowrob:goalLocation rdf:resource="&knowrob;Pose_RQOJGTLU"/>
  <knowrob:startTime rdf:resource="&knowrob;timepoint_1521642520"/>
  <knowrob:taskContext rdf:datatype="&xsd;string">DummyContext</knowrob:taskContext>
  <knowrob:taskSuccess rdf:datatype="&xsd;boolean">false</knowrob:taskSuccess>
  <rdf:type rdf:resource="&owl;NamedIndividual"/>
</knowrob:LiftingAGripper>

<knowrob:TimePoint rdf:about="&knowrob;timepoint_1521642498">
  <rdf:type rdf:resource="&owl;NamedIndividual"/>
</knowrob:TimePoint>

<knowrob:SettingAGripper rdf:about="&knowrob;SettingAGripper_LSGWOPTR">
  <knowrob:bodyPartsUsed rdf:resource="&pr2;pr2_left_gripper"/>
  <knowrob:endTime rdf:resource="&knowrob;timepoint_1521642413"/>
  <knowrob:position rdf:datatype="&xsd;float">0.1</knowrob:position>
  <knowrob:startTime rdf:resource="&knowrob;timepoint_1521642412"/>
  <knowrob:taskContext rdf:datatype="&xsd;string">DummyContext</knowrob:taskContext>
  <knowrob:taskSuccess rdf:datatype="&xsd;boolean">true</knowrob:taskSuccess>
  <rdf:type rdf:resource="&owl;NamedIndividual"/>
</knowrob:SettingAGripper>

<knowrob:Retracting rdf:about="&knowrob;Retracting_VMCPURGE">
  <knowrob:bodyPartsUsed rdf:resource="&pr2;pr2_right_gripper"/>
  <knowrob:endTime rdf:resource="&knowrob;timepoint_1521642395"/>
  <knowrob:goalLocation rdf:resource="&knowrob;Pose_CYTEDZXK"/>
  <knowrob:startTime rdf:resource="&knowrob;timepoint_1521642389"/>
  <knowrob:taskContext rdf:datatype="&xsd;string">DummyContext</knowrob:taskContext>
  <knowrob:taskSuccess rdf:datatype="&xsd;boolean">true</knowrob:taskSuccess>
  <rdf:type rdf:resource="&owl;NamedIndividual"/>
</knowrob:Retracting>
```

(a) Robot NEEMs

```
<owl:NamedIndividual rdf:about="&log;FurnitureStateHalfOpened_vQk8">
        <rdf:type rdf:resource="&knowrob_u;FurnitureStateHalfOpened"/>
        <knowrob:taskContext rdf:datatype="&xsd;string">FurnitureStateHalfOpened-IslandDrawerBottomLeft_WrSn</knowrob:taskContext>
        <knowrob:objectActedOn rdf:resource="&log;IslandDrawerBottomLeft_WrSn"/>
        <knowrob:startTime rdf:resource="&log;timepoint_4.266569"/>
        <knowrob:endTime rdf:resource="&log;timepoint_4.769247"/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="&log;TouchingSituation_D03w">
        <rdf:type rdf:resource="&knowrob_u;TouchingSituation"/>
        <knowrob:taskContext rdf:datatype="&xsd;string">Contact-BowlLarge_E0kQ-IslandDrawerBottomLeft_WrSn</knowrob:taskContext>
        <knowrob_u:inContact rdf:resource="&log;IslandDrawerBottomLeft_WrSn"/>
        <knowrob_u:inContact rdf:resource="&log;BowlLarge_E0kQ"/>
        <knowrob:startTime rdf:resource="&log;timepoint_0.0"/>
        <knowrob:endTime rdf:resource="&log;timepoint_6.7688"/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="&log;FurnitureStateHalfClosed_AdJo">
        <rdf:type rdf:resource="&knowrob_u;FurnitureStateHalfClosed"/>
        <knowrob:taskContext rdf:datatype="&xsd;string">FurnitureStateHalfClosed-IslandDrawerBottomLeft_WrSn</knowrob:taskContext>
        <knowrob:objectActedOn rdf:resource="&log;IslandDrawerBottomLeft_WrSn"/>
        <knowrob:startTime rdf:resource="&log;timepoint_3.51823"/>
        <knowrob:endTime rdf:resource="&log;timepoint_4.266569"/>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="&log;GraspingSomething_0tjr">
        <rdf:type rdf:resource="&knowrob;GraspingSomething"/>
        <knowrob:taskContext rdf:datatype="&xsd;string">Grasp-BowlLarge_E0kQ-LeftHand_xyrY</knowrob:taskContext>
        <knowrob:performedBy rdf:resource="&log;LeftHand_xyrY"/>
        <knowrob:objectActedOn rdf:resource="&log;BowlLarge_E0kQ"/>
        <knowrob:startTime rdf:resource="&log;timepoint_6.7688"/>
        <knowrob:endTime rdf:resource="&log;timepoint_8.422571"/>
</owl:NamedIndividual>
```

(b) VR NEEMs

Figure 5.2: *Two different types of NEEMs. As seen Figure 5.2a, there exist tasks that the robot tries to achieve in robot NEEMs. On the other hand, the physical events and the state changes in environments are recorded in VR NEEMs.*

Although behavioral cloning and inverse reinforcement learning techniques such as (Ng and Russell, 2000; Pomerleau, 1989; Kalakrishnan et al., 2009) are proven to work in different use-cases, they require retraining from scratch even a small change in task descriptions or environmental features. Finn *et al.* present Model Agnostic Meta Learning (MAML) (Finn et al., 2017a) for learning a generic policy that is valid for a set of tasks. MAML tries to find a parameter from which the algorithm can adapt to a given task with few gradient steps. As a follow-up work, Finn et al. (2017b) demonstrate that their MAML-based one-shot imitation methodology can imitate reaching, pushing and placing tasks. Yu et al. (2018) implement a domain-adaptive version using which robots can imitate by observing human demonstrations. For this manner, they have modified the inner-loss function of MAML so that temporal information is also taken into account.

The problem-of-interest in this chapter is how a successful imitation is achieved after observing a single VR demonstration of *newly assigned task* by using a human and robot demonstration dataset from previously executed tasks to build up prior knowledge through MAML.

This section presents the learning methodology. In Subsection 5.4.1 and Subsection 5.4.2, training the meta-model (or prior knowledge) using MAML is described. In the rest, I point out how this meta-model enables the domain-adaptive one-shot imitation learning using gradient updates.

5.4.1 Model Agnostic Meta-Learning (MAML)

MAML (Finn et al., 2017a) enables agents to learn new tasks quickly in a data-efficient way. For training, it requires demonstrations from a set of tasks, i.e., meta-training tasks. MAML assumes that training and test tasks are sampled from some distribution and there exists a meta-parameter set that can be fitted to these tasks by a single gradient update. For this manner, there are two phases, namely the pre-update and post-update phases, for obtaining those meta-parameters.

In the pre-update phase, MAML obtains a meta-parameter draft ϕ_T using the human demonstrations, d^h, and the loss function, $\mathcal{L}$, together with the initialization parameter, θ, and the adaptation parameter, ψ:

$$\phi_T = \theta - \alpha \bigtriangledown_\theta \mathcal{L}_\psi(\theta, d^h) \tag{5.1}$$

where α is learning rate.

In the latter phase, a new pair of θ, ψ is obtained using the formula:

$$(\theta, \psi) \leftarrow (\theta, \psi) - \beta \nabla_{\theta,\psi} \mathcal{L}_{BC}(\phi_T, d^r) \tag{5.2}$$

where β is learning rate and BC is a behavioral cloning parameter that maximizes the probability of successful trials in *the newly assigned task*.

For the domain adaptive applications, Yu et al. (2018) have extended this algorithm as follows. In every iteration, it samples a batch of training task, T. For every task, n human demonstrations, d^h, and a robot execution, d^r, are sampled. In this chapter, I apply the pre-update and post-update phase also for input domain which is different than the original approach.

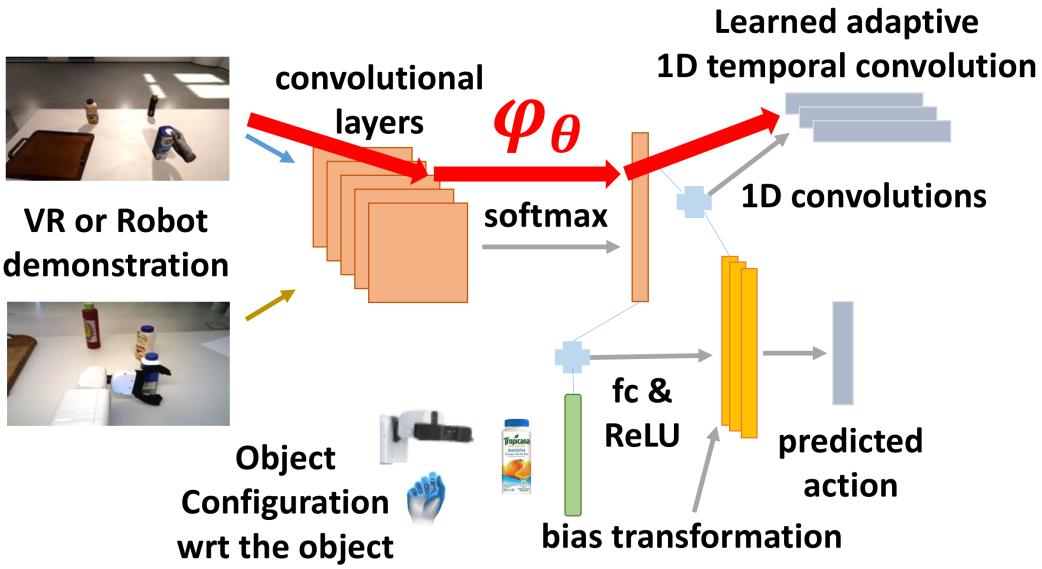


Figure 5.3: *Domain Adaptive Meta-Learning Scheme with VR Demonstrations*

5.4.2 Deep Network Structure

As aforementioned, VR demonstrations (d^h) are recorded inside virtual-reality (Figure 5.3). The input goes through 5 convolution networks with 2, 2, 2, 1, 1 stride in order and concatenated with the position and velocity of the end effector, i.e., the robot's end-effector or the hand of the game character, which are with respect to the target object. Then, the concatenated vector goes through 3 fully connected layers with ReLU activation function for predicting action where the vector is bias-

transformed before the first fully connected layer. The final vector is concatenated with the immediate layer after the softmax layer and goes through three 1D temporal convolution layer to get learned adaptation 1D temporal convolutions.

5.4.3 Bias Transformation

The *bias transformation* technique (Finn et al., 2017b) is used to increase the significance of gradient updates. The importance of the *bias transformation* can be formulated as follows: Let z be the parameter vector concatenated to an output of hidden layer's activation function, x be the output of hidden layer's activation function and y be the input of next layer. If one does not concatenate the parameter vector z to x, y = Wx + b where b is a bias and W is a weight matrix. The gradient of loss with respect to y, $\frac{d\mathcal{L}}{dy}$ is equal to the gradient of loss with respect to the bias $\frac{d\mathcal{L}}{db}$. In this case, the gradient update of the bias b is closely related to the weight W and x.

On the other hand, if the parameter vector z is concatenated to x, y = W_1x + W_2z + b. Here, W_1, W_2 are weight matrices and b is a bias. One can simply interpret $\tilde{b} = W_2\text{z} + b$ as transformed bias and y = $W_1 x + \tilde{b}$. Since $\frac{d\mathcal{L}}{dW_2} = \frac{d\mathcal{L}}{dy} z^T$ and $\frac{d\mathcal{L}}{d\text{z}} = W_2^T \frac{d\mathcal{L}}{dy}$, one gradient update of transformed bias is as below.

$$
\begin{aligned}
\tilde{b}' &= \left(W_2 - \alpha \frac{d\mathcal{L}}{dW_2}\right)\left(\text{z} - \alpha \frac{d\mathcal{L}}{d\text{z}}\right) + b - \alpha \frac{d\mathcal{L}}{db} \\
&= \left(W_2 - \alpha \frac{d\mathcal{L}}{dy}\text{z}^T\right)\left(\text{z} - \alpha W_2^T \frac{d\mathcal{L}}{dy}\right) + b - \alpha \frac{d\mathcal{L}}{dy}
\end{aligned}
$$

Thus, the trained network can easily control the bias with weight W_2 and z.

5.4.4 Temporal Loss

Behavior cloning (BC) is a good fit for imitating the demonstrations of the same agent. However, it is hard to use BC when the demonstrations are from another agent. In order to overcome this, I have adopted a similar temporal loss calculation, as presented in (Yu et al., 2018): Let o_t be an observation at time step t and T be the last time step. Each o_t is mapped to $\pi_\theta(o_t)$ using policy network π_θ. Using multi-layered 1D convolution, $\pi_\theta(o_1) \cdots \pi_\theta(o_T)$ are mapped to a scalar value which reflects the temporal loss (Figure 5.4).

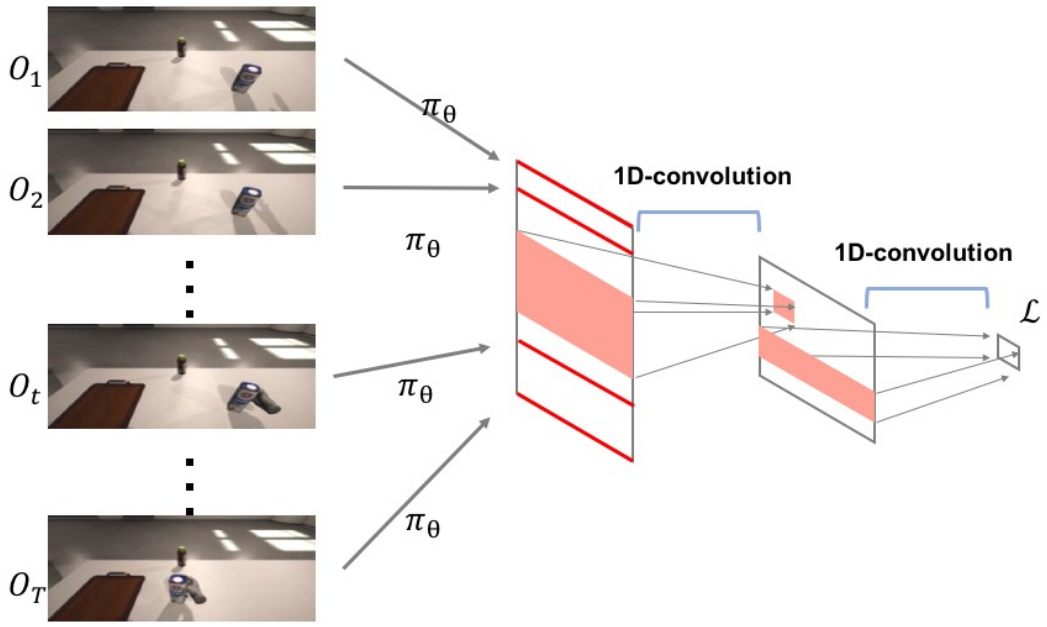


Figure 5.4: *The notion of temporal loss. Each observation o_t is mapped to πo_t. After multiple 1D convolution layers, πo_t are mapped to a scalar $\mathcal{L}$. The training objective is to drive $\mathcal{L}$ to zero.*

The meta-parameter ϕ, which is calculated using temporal loss's gradient during the pre-update phase, is again updated using gradient in the post-update phase. As a result, temporal loss is learned in a way that its gradient helps to decrease loss caused by BC. Moreover, this calculation can detect patterns in the sequential frames because it takes multiple time steps into account using temporal convolution.

5.4.5 Inference of Actions in Newly-Assigned Tasks

This scheme requires only a single VR demonstration of the *newly assigned task*. In that demonstration, the tabletop objects should be identical as real-world settings but the configuration may differ so that the robot does not just replay trajectories but understands the goal and acts accordingly even in a unique configuration. For this manner, an adaptation objective is obtained by computing the adaptive parameter, ϕ_T, as Equation 5.1. This objective, which is a 1D-temporal-convolution, enables the robot not only to recognize the target object but also to adapt actions to a different domain, i.e., from VR to the real world. For the action inference in *newly assigned tasks*, the robot makes use of the same model in Figure 5.3 but it substitutes the input image with the real-time image from the camera. As a result, the parameters

are adopted from the VR demonstration using the calculated objective.

5.5 The Imitation Layer

Figure 5.5 depicts the architecture of the imitation layer. To begin with, the learning process has two stages, namely the meta-learning and the imitation of actions in newly-assigned tasks (see Section 5.4.5 for details). For each training task in the meta-learning, n demonstrations from VR and one demonstration from the robot itself are required. Here what I mean by different *tasks* is different goals for a certain action type. An example would be reaching *Object_A* where *Object_A*, *Object_B*, and *Object_C* are on the table. *Object_A*, *Object_B*, and *Object_C* are located in a random position at the tabletop in each demonstration of this task.

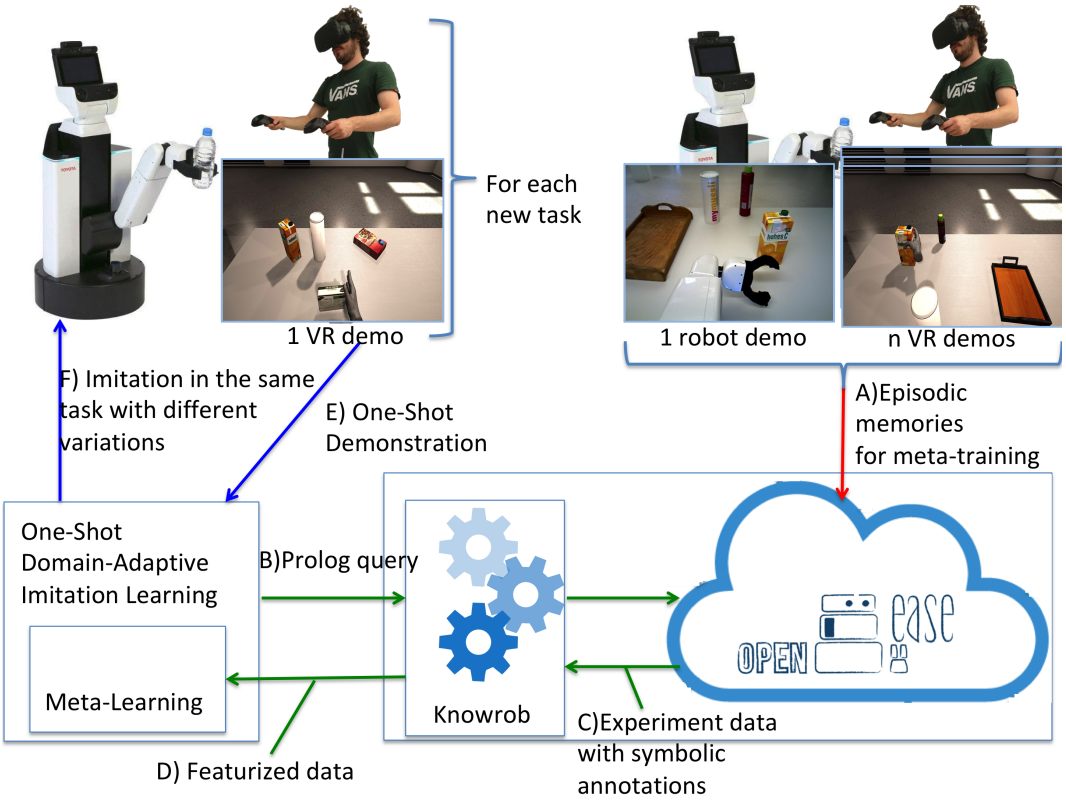


Figure 5.5: *The proposed methodology. The red, green and blue arrows indicate the interaction at the time of game playing, meta-learning and imitation respectively.*

The NEEMs are, then, uploaded to openEASE. During meta-learning, KnowRob, the

local knowledge base, communicates with openEASE using Prolog queries to get training features from the execution logs. In the second phase, one VR demonstration of this new task is supplied. By having this demonstration, the robot is expected to imitate the task in the real world.

In the following subsections, I give more details about the architecture.

5.5.1 VR-Based Game with a Purpose for Kitchen Manipulations

As the modern game engines turn more and more realistic in terms of physics (such as weight, friction, collision, etc.), they become strong alternatives for full-stack simulators like Gazebo. Their most significant advantage is offering these physics capabilities in almost real-time which enables effective human presence in simulated environments.

We (Haidu et al., 2018) present a VR-based serious gaming architecture designated for human demonstrations that uses KnowRob data structures and records execution logs similar to the ones from robots.

In this chapter, I use this architecture to generate a game with a purpose within a virtual kitchen environment that has similar physical properties and a shared object set with the real kitchen in Institute for Artificial Intelligence, Bremen. Figure 5.6 depicts how the game renders photorealistic scenes which make the images generated useful for training deep learning models.

As aforementioned, it takes substantial effort to generate realistic data in the simulator for traditional learning approaches. On the other hand, in this layer, I do not use this game to generate simulated robot data but record human demonstrations in a photorealistic virtual environment to learn a mapping between the robot and the avatar controlled by a human. Therefore, I can use the game data without any modifications. Similarly, the domain-adaptive meta-learning approach can successfully generalize the differences in factors such as weight, friction, and applied force between the real world and the virtual kitchen.

5.5.2 Learning Process

The imitation layer, first, gets the desired features out of NEEMs in openEASE. For this manner, it directs Prolog queries to KnowRob before meta-learning. The Prolog

Figure 5.6: *An example kitchen scene taken from the VR game (left) together with the same scene photographed in the real world (right).*

query that is used for featurizing reaching demonstrations is as follows:

```
?-member(Task, AvailableTasks),
  findall(Features,
   ((entity(Act, [an, action,
        [type, 'Grasping'],
        [bodyPartUsed, Arm],
        [instanceOf, Task]]);
     entity(Act, [an, event,
        [type, 'PhysicalContactEvent'],
        [bodyPartUsed, Arm],
        [instanceOf, Task]])),
     task_start(Act, EndReach),
     StReach is EndReach - 5.0,
     sample_trajectory_positions
        (Arm, StReach, EndReach, Pos),
     sample_trajectory_velocities
        (Arm, StReach, EndReach, Vel),
     append(Features, Pos, Vel)),
  TaskFeatures).
```

This query scans through the instances of *Grasping* actions in robot NEEMs and *PhysicalContactEvent* events in VR NEEMs. I assume that reaching actions start 5 seconds before these. Thus, we sample positions and velocities from the arm trajectories in these intervals. After KnowRob returns the features from the training task set which contains n VR and one robot demonstrations for each task, the layer initiates the meta-learning.

The imitation layer expects a single VR demonstration of the *newly-assigned task*. By looking at this demonstration, it imitates the task in the real-world by adjusting its metamodel with a gradient update of transformed bias (see Section 5.4.5 for details).

5.6 Experiments

The experiments take place in an environment that consists of a breakfast table and multiple kitchen items on it (Figure 5.7). In the real-world setup, a Toyota Human Support Robot (HSR) is used as the service robot platform. In VR, human players interface with the virtual environment using HTC Vive which can track the poses of its controllers and headset in the virtual environment. Due to the lack of full-body tracking, only the game character's hands and his head are tracked in the game.

In order to have photorealistic training images, the physical properties of the tables in the real kitchen and VR are kept the same. Moreover, the objects in VR are rendered using 3D models of real kitchen items (Figure 5.6). I record the image stream with a virtual camera positioned similarly as HSR's RGB-D sensor to simulate its line of sight (Figure 5.7c). However, it is expected that the robot can execute the same task even if the object poses are different in the real-world setup as explained in Section 5.4.5.

In the experiments, the imitation of reaching and pushing actions is tested. For meta-learning, there exist 24 reaching and 12 pushing VR demonstrations for each of 26 meta-training tasks. Additionally, one reaching and one pushing robot execution are recorded for each of these tasks in the real-world settings. These executions are realized using Python scripts. For the imitation of the *newly-assigned tasks*, one VR demonstration is supplied. Position information of the hand (coming from the predicate *sample_trajectory_positions* described in Section 5.5.2) and images (in the form of RGB pixel arrays) are supplied as the features (d). The velocity values returned by *sample_trajectory_velocities* are used as the control signal (ϕ).

For the meta-training, the weight of the inner gradient update is taken as 0.01. For every meta-update, 5 tasks are sampled. Furthermore, the bias transformation is used only on the first layer of the fully connected layers and the bias transformation's dimension is fixed as 20. There exist 4 convolution layers, 3×3 sized filter for convolution layers, number of strides as 4.

5.6.1 Use-case 1: Reaching

The first use-case is the imitation of reaching action. In the game, demonstrators are instructed to touch the object-of-interest without any further specifications. The Python script for generating the robot demonstrations are implemented to reach the center of the object-of-interest's center.

(a) An example real-world breakfast table setting from a pushing task

(b) Robot's eye view of same real-world setting

(c) An example VR-game setting from the same task

Figure 5.7: *Experimental Setup*

There exist 24 VR and 1 robot demonstrations for each of 26 meta-training tasks as mentioned earlier. For testing, 1 VR demonstration of 5 *newly-assigned tasks* is used. For each of these tasks, the robot has executed 8 trials where the tabletop objects were located randomly.

Out of these 40 trials, I have reached 24 and 29 successful executions while using only image features and using both position and image features respectively. These results yield the success rates of 60% and 72.5%.

It is also observed that most of the failures are not because of poor learning performance but rather the physical factors. Firstly, due to the lack of full-body tracking, the game character does not have full arms but only "flying" hands. This limited tracking

Experiment	Only Image	Image & Position
Reach	60%	72.5%
Push	57.5%	77.5%

Table 5.1: *The success rates of experiments*

yields a significant information loss in the training images concerning occlusion. Thus, there are failure cases in which the robot stops reaching because it thinks it already reached the object when its arm is in between the camera and the object (Figure 5.8). Secondly, HSR's arms have only 5 DoF (excluding the grippers) which requires base motions to reach the objects very close or wider away. Such base motions generally can lead to changes in the line of sight. Additionally, it can cause the body to hit the table since I disable the controller's collision avoidance in order not to generate even more base motions.

Figure 5.8: *An occlusion failure during reaching*

5.6.2 Use-case 2: Pushing Left

The imitation of pushing action is the second use-case. For this manner, the demonstrators are asked to push the object-of-interests to the left. They are instructed to grasp-and-then-push for avoiding unintended tilting and falls. In order to generate robot demonstrations, I have implemented a Python script that, first, detects the bounding box of the object-of-interest and, then, reaches the object from its right. After reaching, it closes the gripper and transports the object to the left. Similarly, in the testing, the robot starts to test its pushing action based on the trained model after it reaches the object and closes the gripper.

For this case, there exist 12 VR and 1 robot demonstrations for each of 26 training tasks. As in Use-case 1, 1 VR demonstration of 5 *newly-assigned tasks* is supplied. HSR has executed 8 trials for each training task where the objects were located randomly. After these tests, the success rates were 57.5% and 77.5% when using only image features and using both position and image features respectively.

Similar to Use-case 1, it is observed that many failures have been caused by base motions which change the line of sight and body collision. On the other hand, there were not any occlusion-based failures because grasping the object from its right does not occlude the object.

5.6.3 Discussion

In the experiments, I have demonstrated that the VR-based game with a purpose makes possible to demonstrate tasks to robots without the necessity of being in the same or similar environment. Given that image-only models have lower success rates (Table 5.1), one can also claim that having demonstrations from a game environment which the loggers have access to ground truth boosts the learning rate. On the other hand, there are still limitations, the most significant one is not having full-body tracking in the VR setup. Although our lab infrastructure has Optitrack tracking system which can interface with the virtual reality, I have not made use of it in the experiments. As one of the aims being crowdsourcing, I want to use only the devices that are affordable and reachable by consumers. I foresee that up-coming virtual-reality products such as HTC Vive Tracker (a marker-based tracker add-on to HTC Vive) will enable us to solve this limitation to a certain degree.

Another limitation is having a simple robot arm that requires base motions during manipulations. Toyota HSR platform for this work was chosen as the robot platform because this research is conducted within the German-Korean Research Project (GenKo) and HSR is the only common robot platform with manipulation capabilities between Biointelligence Lab (German-Korean Research Project Partner) and the Institute for Artificial Intelligence. Although it is a suitable platform for many household tasks such as ones in Robocup@Home context, its manipulation capabilities are limited without the mobility of its base. Thus, I believe that using a more sophisticated robot arm such as Franka, UR-5 or PR2 would substantially boost the performance.

As a side note, I have made the implementation[1] and the training data[2] publicly accessible for promoting *open research*.

[1] https://github.com/asilx/one-shot-neem-training
[2] https://www.dropbox.com/s/jmb27b1rrg54kqm/crowd_source_im_lrn.zip

Chapter six

Learning using Prospection

This chapter is about enabling robots to simulate actions to find parameterizations for the desired outcome. The research and scientific results described in this chapter is also presented in (Bozcuoglu and Beetz, 2017).

6.1 Prospection for Robots

A common point of the complex household tasks is that they are, usually, under-specified and some context-specific knowledge and learning is needed to perform them successfully. Humans frequently prospect in order to predict outcomes just before carrying out the actual tasks. These predictions guide them to parametrize their actions correctly with-respect-to factors such as speed, positioning, and applied force (Druckman et al., 1992). A possible way to have this mental ability in a robot is interfacing with a simulation system in which it can represent the world model, self-abilities and can operate in this simulated environment with its own control and planning systems. Such simulations are also referred as *mental simulations* (Pham and Taylor, 1999).

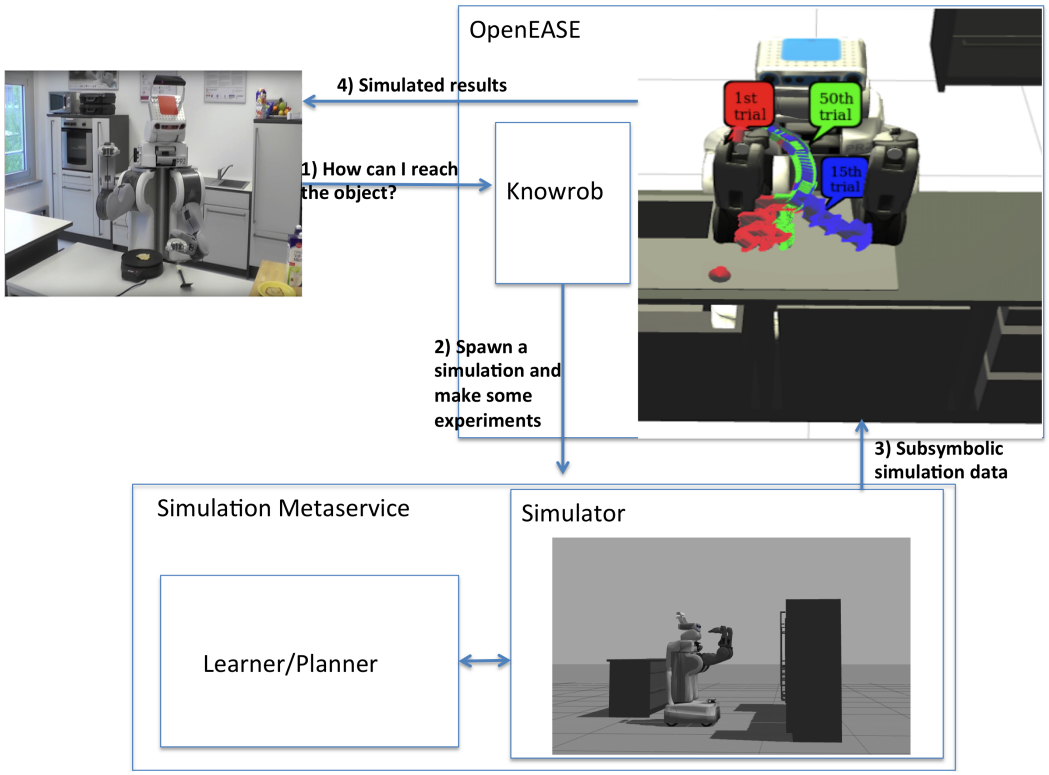


Figure 6.1: *An example activity diagram of the proposed layer in which PR2 robot asks how to reach an object. The layer makes some reaching trials in the simulation and returns results via openEASE.*

On the other hand, the robot's own processing power or storage capacity is limited for employing heavy simulation processes. This is the point where the cloud robotics platforms come to help and offer remote storage and processing services. These platforms such as openEASE (Beetz et al., 2015) can host a prospection service which goes beyond the traditional knowledge acquisition from other sources or previous experiments by supplying knowledge about what may happen if the robot applies a course of actions in the future.

In this chapter, I present such a service inside within the proposed learning framework as the prospection layer. This layer has the following features.

1. people and robots can interface with this layer using Prolog queries for the world model (and the state), the agent capabilities and the problem.
2. the layer applies learning for the given parameters and returns results
3. these results can be reasoned later on by using other components of openEASE since they share similar data structure with the rest of openEASE modules.

An example activity diagram of the proposed system is illustrated in Figure 6.1.

6.2 Prospection Layer

6.2.1 Subsystem Overview

Prospection layer enables robots to do followings:

1. describing a world, a state, and a problem such as *"PR2 is at Point-A inside Kitchen-B and it wants to reach Cup-C."*.
2. choose number of trials and learning algorithm such as *"For this problem, run SARSA algorithm with 50 trials and give me the learned trajectories"*.
3. reason about the results such as *"What was the less time-taking trajectory in these 50 trials?"*.

Initially, the robot describes the environment and asks the learning problem formulated as a Prolog query to openEASE which, then, spawns a Gazebo simulation (Koenig and Howard, 2004) and, then, initiates the learning process according to given parameters. During this process, the simulation is recorded by a NEEM logger. In the end, the resulting NEEM is directly transferred to openEASE for further reasoning. The overall system architecture can be seen in Figure 6.2.

6.2.2 Learning Framework

I designate the primary use-case as how to achieve the goal using reinforcement learning (RL), although this prospection service is also suitable for other purposes such as motion planning, robot control and obstacle avoidance. In order to access RL algorithms within the layer, I make use of *The Brown-UMBC Reinforcement Learning and Planning* (BURLAP) (MacGlashan, 2016) which offers developers to a broad range of reinforcement learning and planning algorithms with a nice API and ROS bridge. Once the robot chooses an algorithm from BURLAP and gives necessary learning parameters to the system with the respective predicates, it can start the simulation and the communication between BURLAP and the simulation is created automatically using ROS topics and services.

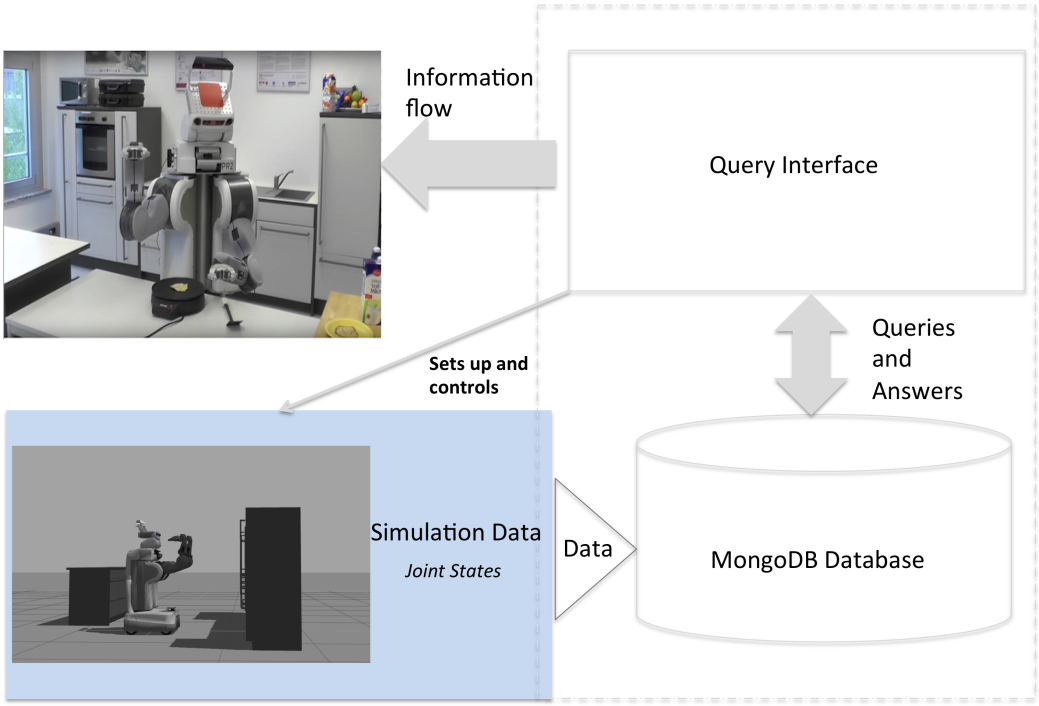


Figure 6.2: *The proposed prospection layer's architecture.*

6.2.3 How to define the environment and the goal of the simulation?

Similar to the other layers, some prolog predicates (shown in Table 6.1) are introduced in order to interface with the prospection layer. First of all, *choose_map*/2 is used for loading the semantic map of the environment. As aforementioned, if the user wants to use new environments in simulations, she can create new semantic maps (The semantic map format used in openEASE is presented in (Pangercic et al., 2012)).

Using the predicate *define_goal*/3, the simulation goal is declared. This goal can be, in the current implementation, *reach, grasp, traverse*. *Reach* is used for getting the arm into the given point and *grasp* is for grasping the specified object. Finally, *traverse* is for getting the robot body to the given point.

With the predicate *start_simulation*/3, the simulation is initiated using the parameters that have been defined. Finally, *import_tf* migrates the data which has been recorded during the simulation to openEASE's database.

Predicate	Description
choose_map(SemanticMap, ParameterList)	Sets semantic map to the given map. Returns learning parameters as a list.
define_goal(Goal, Offset, ParameterList)	Sets the goal (with inner parameters) and offset. Returns learning parameters as a list.
choose_state_factory(StateClass, ParameterList, JavaInstance)	Sets the state factory for RL with the given class name and parameters. Returns generated state factory.
start_simulation(ParameterList, algorithm(Alg), NoOfTrials)	Starts simulation with given parameters, algorithm and number of trials.
import_tf(ListOfTrials, Path)	Imports the simulation data to openEASE.

Table 6.1: *Predicates for interfacing with the simulations.*

Predicate	Description
use_simulation_data(ListOfTrials)	Sets data that will be used for reasoning.
visualize_trajectory(TrialName, Link)	Visualizes the trajectory of the given trial and the given robot link
trajectory_duration(TrialName, Duration)	Returns the duration time of given trajectory.
trajectory_length(TrialName, Link, Length)	Returns the length of the given link's trajectory from start point to end point.

Table 6.2: *Predicates for reasoning on performed simulations.*

6.2.4 How to reason on performed simulations?

The whole reasoning process takes place on server-side so that the clients do not need to download, install, use any kind of code or data and interface with it by their own processes.

For this purpose, I have introduced another set of predicates as shown in Table 6.2. The first predicate in this table is $use\_simulation\_data/1$ which sets the simulation data that will be used. Secondly, $visualize\_trajectory/1$ visualizes the given trial's trajectory in the openEASE canvas. Finally, $trajectory\_duration/2$ and $trajectory\_length/2$ returns the duration and length of the given trial respectively.

6.3 Experimental Setup

This layer is showcased in a scenario where PR2 accomplishes household tasks in the kitchen of Institute for Artificial Intelligence (IAI) (Figure 6.3). The experimental setup involves a cylindrical object being placed on the counter. PR2 is equipped with only the object location and it needs to first approach the object and, then, reach it with its right arm.

In the first case, PR2 tries to traverse a location next to the object location and, in

Figure 6.3: *The kitchen setup that PR2 operates.*

the second use case, it tries to reach it. I have used *State-Action-Reward-State-Action* (SARSA) algorithm (Rummery and Niranjan, 1994; Singh and Sutton, 1996) for both cases.

6.4 Use Cases

6.4.1 How to traverse to a given location?

As the initial use-case, I consider a task in which PR2 naively tries to go to the given location so that it can do some manipulations with the object-of-interest. For this task, SARSA algorithm is applied with the following reward function:

$$rwd(S_{curr}, S_{nxt}) = \begin{cases} 5000, \\ \text{for } d_{goal}(S_{nxt}) \approx 0 \\ \\ -5000, \\ \text{for } execution\_time > timeout \\ \\ d_{goal}(S_{curr}) - d_{goal}(S_{nxt}), \\ \text{for } d_{goal}(S_{curr}) \neq d_{goal}(S_{nxt}) \end{cases}$$

where $d_{goal}(S)$ gives the distance to the goal location.

In order to describe the problem with the world and robot model, I, first give the semantic description of PR2 and, then, choose IAI Kitchen semantic map as the world. Finally, I define the goal as traversing to the given coordinate point:

```
?-owl_parse('pkg://knowrob_srdl/owl/PR2.owl')
   choose_map(
    'pkg://semantic_maps/kitchen_pr2.owl',P),
   define_goal(traverse(0.5, 0.05,
       0.0), O, JListMap).
```

After this step, the experiment can also be observed in openEASE canvas (Figure 6.4).

As the next step, the simulation is started with the desired learning algorithm and parameters:

```
?- choose_state_factory(
      'SimpleHashableStateFactory', [], HF),
   start_simulation(HF,
       algorithm('SarsaLam'), 10).
```

After the simulation process ends, the corresponding NEEM is transferred to openEASE and available for reasoning (Figure 6.5).

```
?- Trials = ['tf0', 'tf3', 'tf7', 'tf9'],
   use_simulation_data(Trials),
   forall(member(T, Trials),
   visualize_trajectory(T, '/base_footprint')).
```

As seen from Figure 6.5, the generated trajectories are getting shorter over time. In the last trial, the robot has made a relatively direct approach to the desired position.

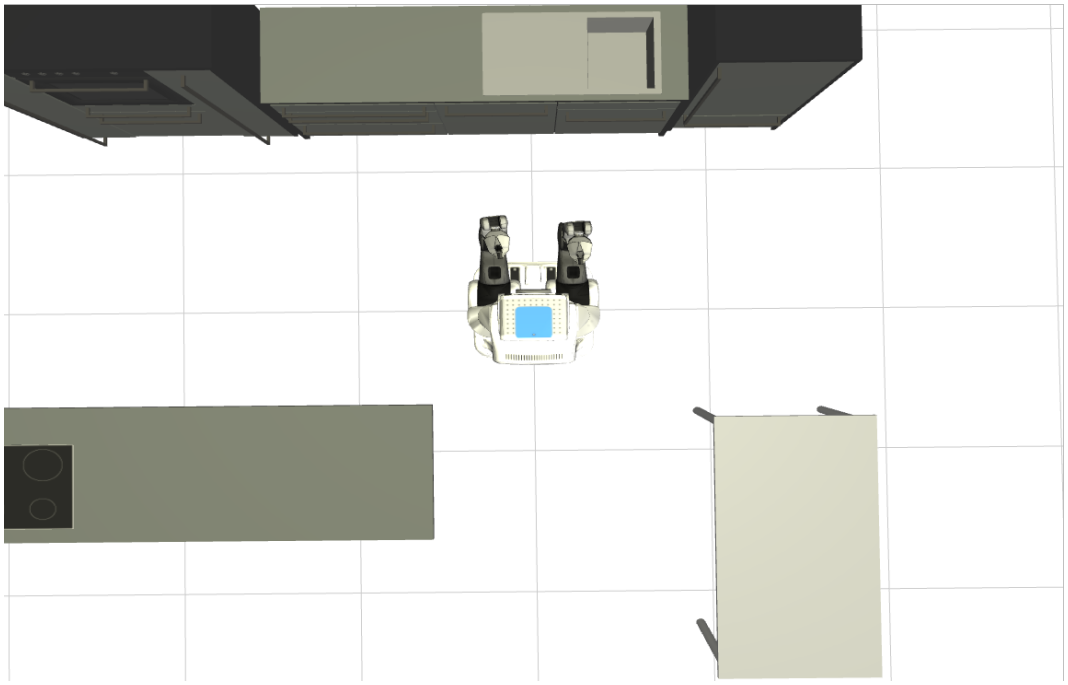


Figure 6.4: *The openEASE canvas after defining the world and robot model.*

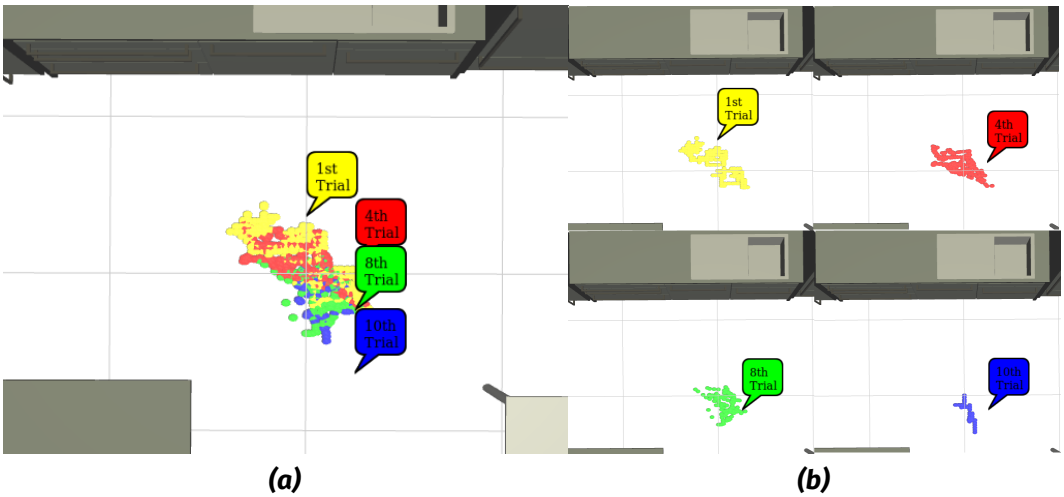


Figure 6.5: *The visual results of Section 6.4.1. The left image* (A) *shows the selected trajectories all together and the right image* (B) *shows them seperately for better visualization.*

6.4.2 How to reach an object?

In the second case, PR2 has already approached the object and, as the next step, prospects on reaching it. For this manner, I apply SARSA algorithm with the following reward function:

$$rwd(S_{curr}, S_{nxt}) = \begin{cases} 10000, \\ \text{for } d_{goal}(S_{nxt}) \approx 0 \\ \\ -10000, \\ \text{for } execution\_time > timeout \\ \\ d_{goal}(S_{curr}) - d_{goal}(S_{nxt}), \\ \text{for } d_{goal}(S_{curr}) \neq d_{goal}(S_{nxt}) \end{cases}$$

where $d_{goal}(S)$ gives the distance to the goal object.

The simulation environment, the simulated robot and the learning problem are set with the following query:

```
?-owl_parse('pkg://knowrob_srdl/owl/PR2.owl')
    choose_map(
      'pkg://semantic_maps/kitchen_pr2.owl',P),
    define_goal(reach(-0.95, 0.67,
        0.83), O, JListMap).
```

After that, the resulting world and robot model as well as the object-of-interest in the given position is visualized in openEASE (Figure 6.6A).

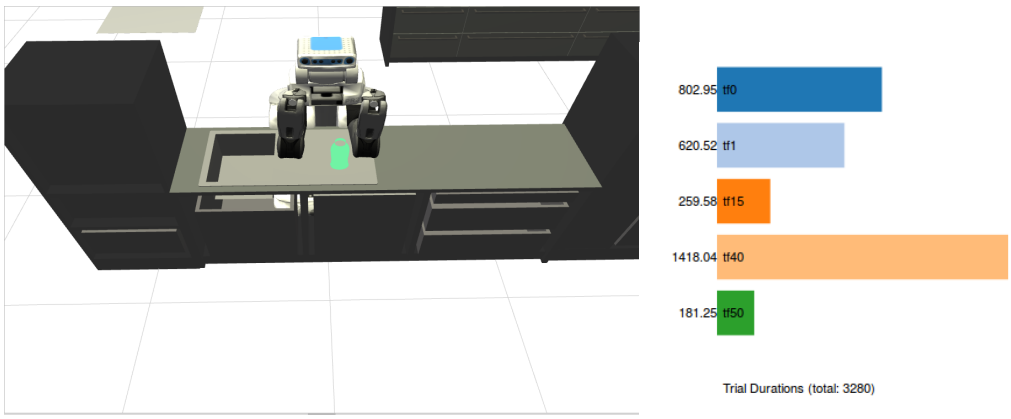
Next, openEASE executes the simulation to run with SARSA and 50 trials:

```
?- choose_state_factory(
       'SimpleHashableStateFactory', [], HF),
    start_simulation(HF,
        algorithm('SarsaLam'), 50).
```

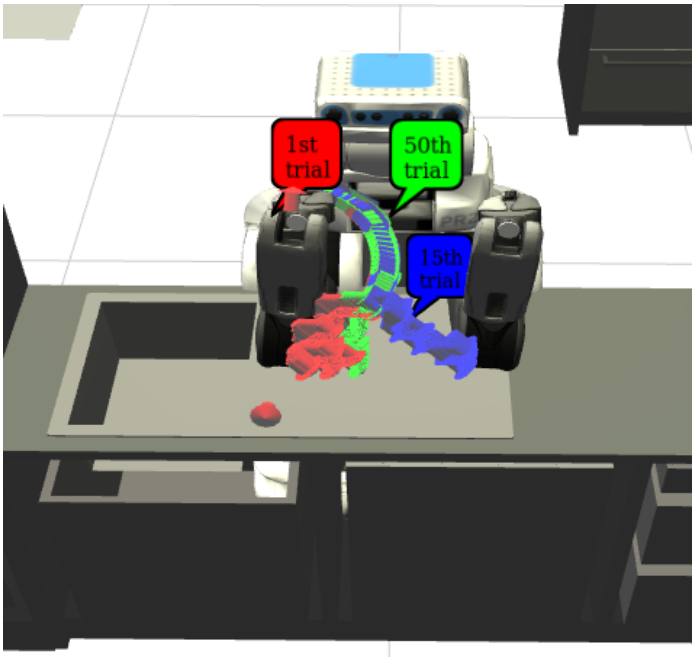
At the end of the simulation process, the trajectories inside the resulting NEEM can be visualized trajectories some of the trials with using the predicate *visualize_trajectory*/1 (Figure 6.6C).

By having the trajectories available, one can also compare and analyze them with respect to different factors. For example, using the following query, it is possible to analyze the duration of the selected queries. Using the chart visualization tool of

openEASE, humans can also see the result as a bar chart (Figure 6.6B).



(a) (b)



(c)

Figure 6.6: *The visual results of Section 6.4.2. The left image* (A) *shows the openEASE canvas after defining the world and robot model. The image* (B) *depicts a bar chart which shows trajectory durations in seconds. Finally, the image* (C) *visualizes trajectories of various trials.*

```
?- Trials = ['tf0', 'tf1', 'tf15', 'tf40'
            , 'tf50'],
   use_simulation_data(Trials),
   findall(Tf-D,
           (member(Tf, Trials),
           trajectory_duration(Tf, D)),
   Pairs),
   show(Pairs).
```

As illustrated in Figure 6.6B, reaching the object takes less and less time after each trial. There is only one exception for this which is the 40th trial. This trial took much higher time than all the others and since the threshold for this experiment was set to 1400 sec, it is the case that the robot could not reach the object in this trial. Being able to reason about such failures after learning will lead to better use of learning itself. For example, if the number of trials had been set to 40 instead of 50, the robot would have identified that the duration in the last trial was a failure (due to the fact that the duration is above the threshold) and would have looked for other trajectories to execute.

6.5 Discussion

With the prospection layer available, robots can delegate *computationally-expensive* simulations to more powerful computing services. In addition, I have implemented an SWI-Prolog library so that the developers and robots can describe the world state, their goals, and the problem. Using RL algorithms inside BURLAP framework, robots can learn how to apply actions. By recording NEEMs of the simulated agent, there is also a possibility to reason about the simulations. Taking the simulated trials into account, the agent can reason about in *which* trial it reached the goal in a shorter time period and *what* was the shortest trajectory among these trials.

This ability is a powerful mechanism for intelligent robots to identify failures and false-positives in the learning process. In particular, for reinforcement learning, it is not always the case that the generated trajectory is better than the previous ones. Using the additional parameters that a simulated environment can supply, the robot can assess this kind of issues and make better use of learning.

To conclude, the idea behind developing this layer is that one can employ much more processing power in a cloud system than robots' own computing facilities. On the other hand, the performance gain will be depending on the hardware.

Part III

Learning Manipulations from Past Robot Experiments

Chapter seven

Affordance Modeling

Accomplishing complex manipulation actions relies highly on machine learning models which are trained with features that are environment- and agent-specific. In order to successfully adapt actions to different agents and environments, these models should be updated all the time. Thus, keeping these models in a cloud platform where other agents can also access and change them is a prominent way to exchange and reuse these models. In this chapter, I provide such a methodology and test it in a scenario where a PR2 executes *opening a fridge* action.

7.1 Affordances and Intelligent Manipulation

One of the core concepts in ecological psychology coined by Gibson (1986) who argued that animals perceive their action affordances not isolated from the environment they live. In other words, affordances are extendable by practicing, but the environmental features and physical laws constrain these improvements. Such explorations over actions and their consequences would be useful for robots to improve their action executions. Along with this idea, the development of the notion of *affordances* for robots has been well-studied subject with different use cases such as traversability of mobile robots (Uğur and Şahin, 2010; Bozcuoğlu and Şahin, 2011), grasp-ability of different objects (Detry et al., 2011; Ugur et al., 2012), and human-robot-interactions in social context (Moratz and Tenbrink, 2008).

In this dissertation, I anchor the concept of *affordances* with the action parametriza-

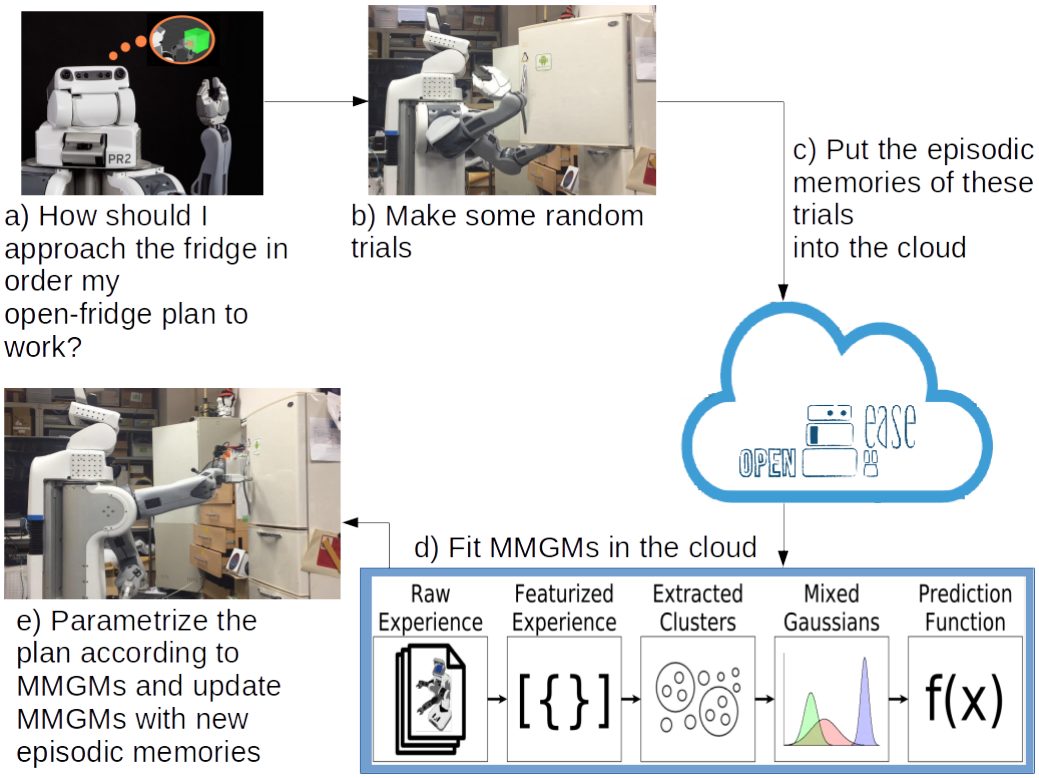


Figure 7.1: *The architecture of the proposed methodology from the robot's perspective. Prediction function, f_x, denotes Eq. 6 given in this chapter.*

tion and present a methodology for multi-modal affordance analysis where robots sample parameters for their actions from the respective Gaussian mixture models (GMM). In return, they update these models with every execution according to its outcome, i.e., success or failure. In other words, I do not particularly deal with explorations of affordances as in (Uğur and Şahin, 2010; Bozcuoğlu and Şahin, 2011; Detry et al., 2011; Ugur et al., 2012) but I assume that the robot can afford the action in the current circumstances. Hence, it should model which motion parametrization leads to a successful action.

Implementing this methodology as a part of openEASE also enables robots to use others' affordance models and, also, to access each other's experiences while modeling. Thanks to NEEMs, there exists not only use case-specific subsymbolic data for our parametrization learning method but as a complete "brain dump" from the corresponding execution which contains semantic high-level knowledge, such as what kind of action was carried, which arm was used, what the goal was, and if there exist any failures, together with all existing subsymbolic data like robot trajectories, joint-states and sensory inputs.

The pipeline is shown in Figure 7.1. Initially, two datasets, one for successful action executions and one for failed ones, are obtained by querying openEASE. After that, the method fits multivariate Gaussian mixture models (GMM) to both datasets. The robot uses these models for the generation of a parameter prediction function. New action executions are used for updating the existing model.

7.2 Obtaining Plan Parameters from the Episodic Memories

Section 2.3.4 lists a set of predicates that can be used for reasoning and accessing context-specific data about past experiences. Together with the standard KnowRob predicate library for inferring environmental and self-information, accessing the experimental knowledge and making mathematical operations, these predicates offer a solid base to implement the data-set-builder queries for affordance modeling.

For example, the query below asks all action instances whose context is *OpenFridgeDoor* in the symbolic knowledge base. Next, it filters out the negative ones with *task_success(Act, true)*. Looking at the start of each action, it asks the robot's pose information with respect to the global reference frame to the non-symbolic database. After that, this pose information is transformed into the reference frame of the fridge door and saved as 7D pose information (3 dimensions for representing the position and 4 dimensions for quaternion).

```
?-findall(FeatureList,
      (entity(Act, [an, action, suboptimal
        ['task_context', 'OpenFridgeDoor']]),
      task_start(Act, StartTime),
      task_success(Act, true),
      belief_at(robot('base_link', RobotPose),
        StartTime),
      pose_into_relative_coord(RobotPose,
        FridgePose, RelativePose),
      matrix_translation(RelativePose,
        RelTrans),
      matrix_rotation(RelativePose, RelRot),
      append(RelTrans, RelRot, FeatureList)),
  FeatureListofLists),
  generate_feature_files(FeatureListofLists,
   'positive.csv').
```

7.3 Clustering Training Data

NEEMs contain parameters and pose information with respect to a global reference frame. To make use of this data in context-specific learning applications, a representation in the local reference frame is often more useful. With such a representation, users can easily reuse or adapt the existing models to different modalities. For instance, if there exists a door-opening model with respect to its hinged joint, one can reuse this model for a different door. In the presented experiments, this is the reference frame of the fridge door. The query in Section 8.3.1 converts PR2's 6D pose at the beginning of *OpenFridgeDoor* task from the global reference frame to the frame of the fridge door.

Experiments with random parameterizations show that it can open the fridge not with uniformly-distributed poses but rather in different subzones such as approaching from the right with modest proximity or approaching from the left with less proximity. Thus, given that these poses are scattered throughout the area, one can cluster them according to their feature vectors.

In order to come up with a reasonable number of clusters, the dataset is clustered using K-Means up to a predefined maximum cluster number. For each clustering candidate, its average point silhouette value (Rousseeuw, 1987) is calculated. As a result, the candidate which has the lowest silhouette value is chosen as the clustering for modeling.

7.4 Fitting Multivariate Gaussian Mixture Models (GMMs)

Using the query shown in Section 8.3.1, the positive and negative datasets are obtained. Each dataset is divided into clusters using silhouette value analysis.

For each cluster, the method fits a multivariate Gaussian model whose covariance matrix C_i is given below.

$$C_i = \sum_{k=1}^{n_i} \left(X_{i,k} - \overline{X}_i\right)^T \left(X_{i,k} - \overline{X}_i\right) \tag{7.1}$$

where i and n_i denote the index of the cluster and the size of the cluster respectively. Since training datasets can get large over time, the method keeps only the covariance

matrix $C_i \in \mathbb{R}^{p \times p}$ and the mean $\overline{X}_i \in \mathbb{R}^p$ of clusters in the working memory. The density function $f_i(X)$ for obtaining the distribution of test data $X \in \mathbb{R}^p$ for the MGM of cluster i is calculated as:

$$f_i(X) = \exp\left(-\frac{1}{2}\left(X - \overline{X}_i\right)^T C_i \left(X - \overline{X}_i\right)\right) \tag{7.2}$$

In the end, there exists one GMM for the positive dataset and one GMM for the negative dataset by applying equal weight $\frac{1}{n}$ to each of the n clusters involved. In order to sample from their distribution, I evaluate the mixture's positive and negative density functions:

$$F_{\{p,n\}}(X) = \sum_{i=1}^{n^{\{p,n\}}} w_i^{\{p,n\}} f_i^{\{p,n\}}(X) \qquad w_i^{\{p,n\}} = \frac{1}{n^{\{p,n\}}} \tag{7.3}$$

where p and n denote *positive* and *negative* respectively.

Thanks to their efficiency and speed, these calculations can be repeated any time on demand or arrival of new datasets.

7.5 From Affordance Models to Prediction Function

After GMMs are stored as data assets in the knowledge base, a prediction function from these models is derived for the robot to query during the executions. This function is used by the robot for inferring the best possible position at the execution time. Since robots update these functions after every new execution, the following steps towards generating prediction functions are defined:

- By having two GMMs one from the positive dataset and one from the negative dataset, the methodology first identifies bounding boxes of these datasets in the feature space and takes the union of these as the prediction function domain.
- With a predefined step-size, it scans through the success and failure probabilities of each step.
- It takes the weighted average of these two GMMs according to the number of positive and negative training samples.

$$F_{likeliness}(X) = \frac{n^p F_p(X) + n^n(1 - F_n(X))}{n} \tag{7.4}$$

- With the same step-size, it scans through the success and failure probabilities of each point and registers ones that are the most likely to succeed, i.e. *global_maximas*.
- A multivariate Gaussian model (MGM) is fitted on this *global_maxima* set.
- Using the mean and covariance of this MGM, a general prediction function is obtained:

$$f_{mx}(\mathrm{X}) = \exp\left(-\frac{1}{2}\left(\mathrm{X}-\overline{\mathrm{X}}_{mx}\right)^{T}\mathrm{C}_{mx}\left(\mathrm{X}-\overline{\mathrm{X}}_{mx}\right)\right) \tag{7.5}$$

$$\mathrm{C}_{mx} = \sum_{k=1}^{n_{mx}}\left(\mathrm{X}_{mx,k}-\overline{\mathrm{X}}_{mx}\right)^{T}\left(\mathrm{X}_{mx,k}-\overline{\mathrm{X}}_{mx}\right) \tag{7.6}$$

The predicate that follows these steps to generate the prediction function is *get_likely_pose/3* (see Table 7.1) which accepts the current robot pose and feature files as the parameters and returns the success probability along with the mean position which implies the point with the highest probability. If the current pose does not have a significant probability, the robot moves to this point.

This methodology, in essence, brings two main advantages. First, it enables us to model affordances based on every trial where the robots can also update the model with the negative samples to reflect their certainty of failure. Second, these prediction functions have a single multivariate Gaussian model (MGM) which makes sampling easier than mixed Gaussian models.

7.6 Integration of Affordance Models with the Symbolic Knowledge

For a KnowRob-based implementation, there is no need for any additional mechanism or layer for interfacing. Figure 7.2 depicts how *opening-container* affordance models together are hooked up with the existing taxonomy. Using special Prolog routines called *computables* (Tenorth and Beetz, 2013), the information coming from affordance models are asserted to the symbolic knowledge on-demand. When the robot asks a query like *"How should I approach to the fridge for opening it"*, the computable predicate, *desired_pose*/2, is executed with the fridge instance as the parameter. As a result, this routine returns a pose calculated using *Equation 6* and asserts this to the symbolic knowledge base. This pose instance is used by the robot control executive as the goal pose.

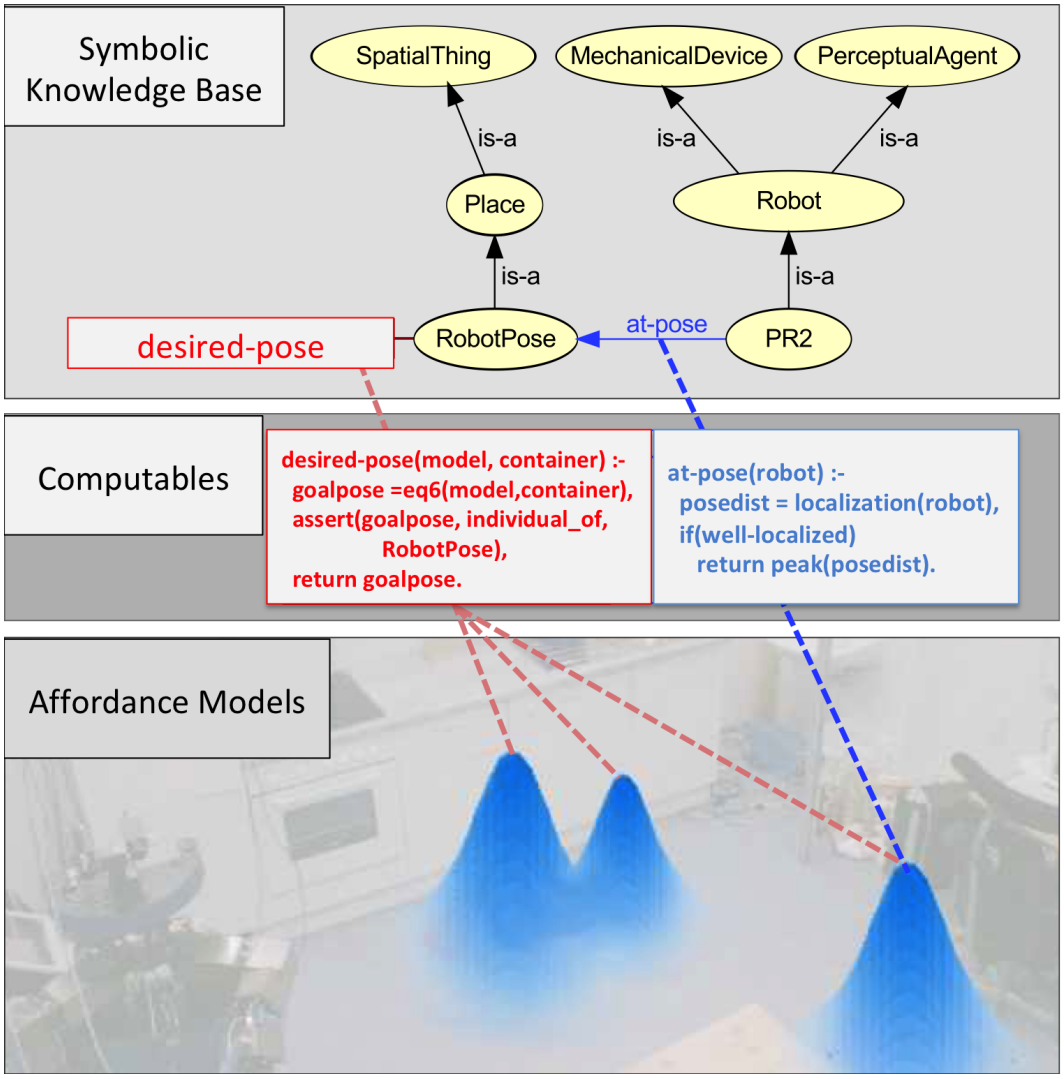


Figure 7.2: *An example of how the affordance models and the robot's localization routines are represented on-demand in the symbolic knowledge base using computable procedures.*

In order to enable users to interface with Gaussian mixture modeling and silhouette value-based K-Means clustering analysis, I have provided the predicates listed in Table 7.1.

7.7 Experimental Setup

In the experiments, a PR2 tries to open a fridge in a kitchen environment (Figure 7.4). For this manner, it parameterizes the offsets for positioning itself in front of the door

Predicate	Description
mixed_gaussian(PosFile, PosCluster, NegFile, NegCluster, Gauss)	Given the positive and negative features file (*PosFile*, *NegFile*) and maximum number of clusters (*PosCluster*, *NegCluster*), this predicate first clusterize the negative and positive datasets, then, fits multi-variate Gaussian models for each. At the end, it stores the final GMM to the variable, *Gauss*.
get_likely_location(Gauss, Mean, Cov)	This predicate search through the global maxima in the GMM. After finding them, it fits a multivariate Gaussian model (MGM) to these maxima and return its mean and covariance. These properties are used by the robot during the execution time for choosing an appropriate location.
generate_heat_maps(Gauss)	This predicate is used to visualize the generated GMM whose path is given in the variable, *Gauss*.

Table 7.1: *Predicates that are implemented for generating multivariate Gaussian mixture models (GMM).*

using the presented methodology and executes an opening action from that location.

After each trial, PR2 automatically labels the task as success or failure according to the detection of the milk box inside the fridge after opening. The planer waits for a successful perception task since I have observed a lot of near-miss opening-fridge actions in which the robot actually opens a fridge but it cannot perceive or manipulate inside of the fridge. Since these attempts do not serve the goal, they are classified as negative samples.

As the features, Euclidean distances with respect to fridge's reference frame (as illustrated in Figure 7.4) in X- and Y- axes are used.

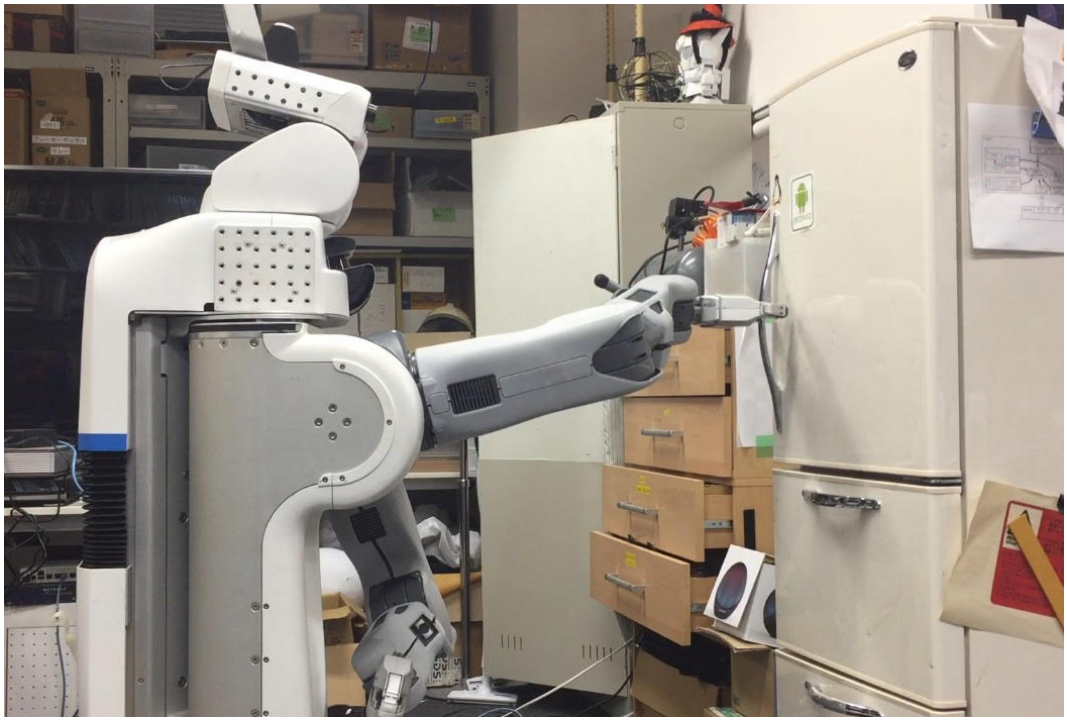


Figure 7.3: *PR2 opening the fridge door in the kitchen where the experiments are carried on.*

7.8 Experiments

PR2 bootstraps the GMMs with an initial set of 7 episodes in which it tries to open fridge with random parameters until it succeeds. The random values are taken in the interval of $[-1.5, 0]$ for X-axis and in the interval of $[-1, +1]$ for Y-axis.

This initial set of episodes contains in total 7 positive and 18 negative door opening trials. As seen in the attachment video, the failures are caused by factors such as hitting self while reaching the door handle and slipping the door handle from gripper while opening. After initial randomized training set for bootstrapping the affordance model is collected, the robot generates the initial version of the cost function and uses it to position itself in front of the fridge. There exist 8 episodes of this kind in which there exists 7 positive and 5 negative trials.

In the subsections, I begin by describing the results for positive GMM-only with 7 and 15 episodes. Then I give the models in which both GMMs are taken into account. Using these models, I can analyze how the system scales with more data and how negative trials improve the overall performance.

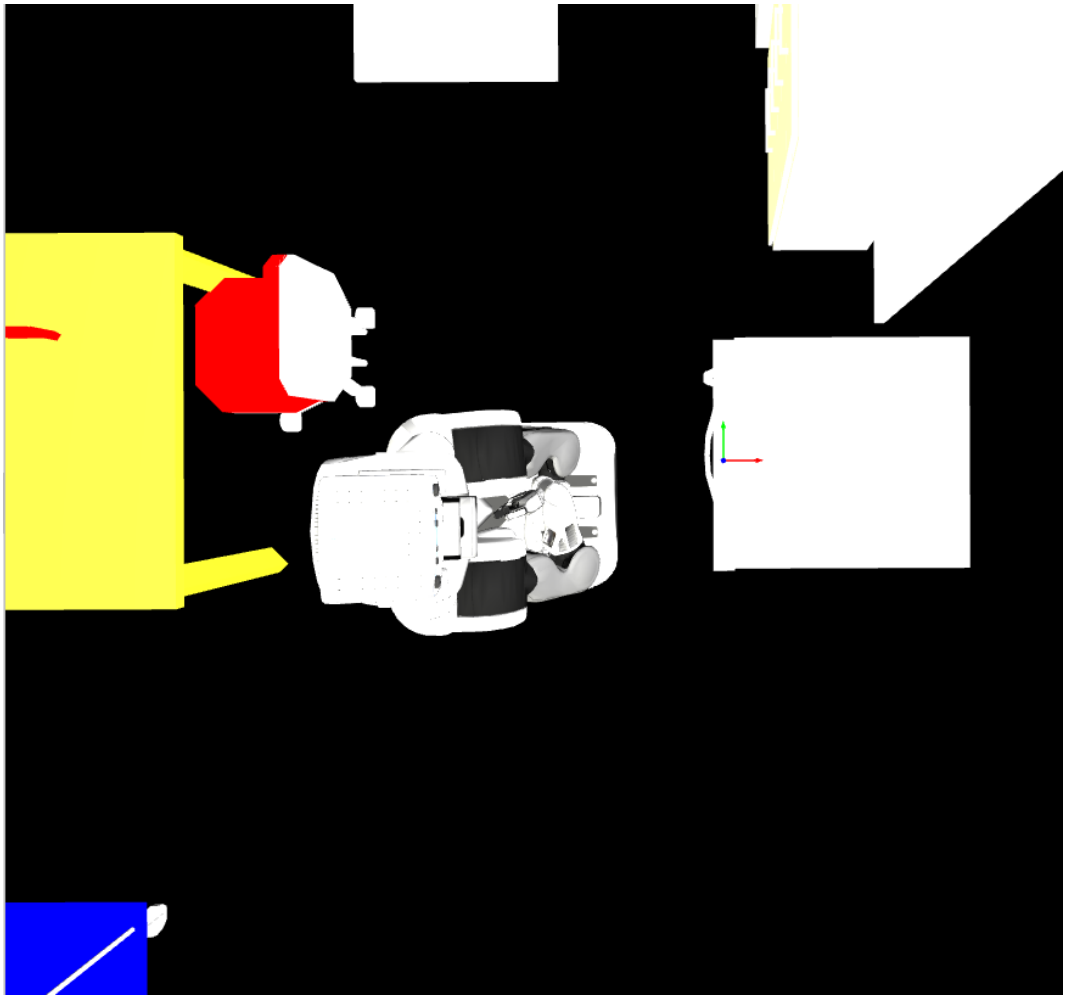
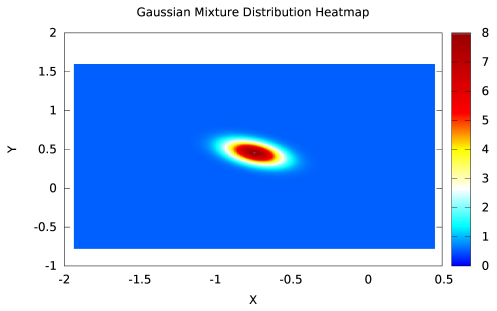


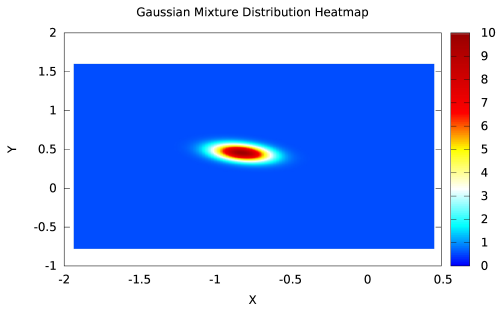
Figure 7.4: *A top-down view from 3D Visualization canvas of openEASE web Interface. Fridge's reference frame is also visualized in the scene where the red, green, blue axes correspond to X-, Y-, and Z-axes respectively.*

7.8.1 Positive GMM using Trials from 7 Episodes

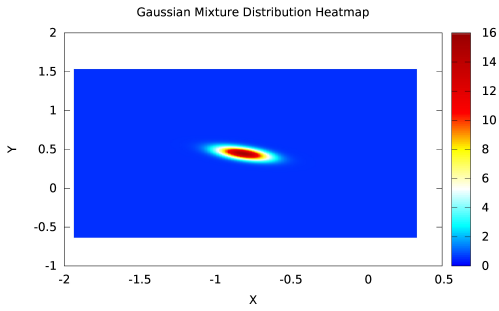
I start with the model GMM generated with positive trials only. For this case, the maximum cluster parameter is given as 3 to silhouette-based K-Means cluster analysis. As shown in Figure 7.5a, this GMM generalizes a relatively large region-of-interest in this case without much constraints with respect to the proximity to the fridge reflected the heat map.



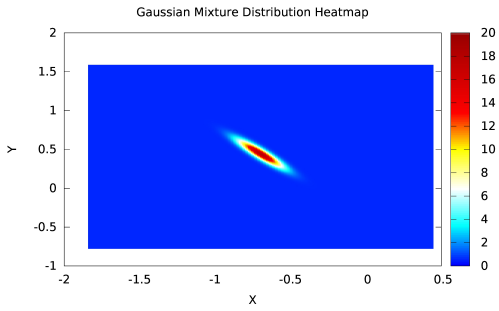
(a) Positive GMM created using trials from 7 episodes



(b) Positive GMM created using trials from 15 episodes



(c) Positive and Negative GMMs created using trials from 7 trials



(d) Positive and Negative GMMs created using trials from 15 trials

Figure 7.5: *Heatmaps generated from the cost functions of the use-cases described from Subsection 7.8.1 to Subsection 7.8.4 The X- and Y- frames are in meters and with respect to the fridge's reference as seen in Figure 7.4*

7.8.2 Positive GMM using Trials from 15 Episodes

The region of interest becomes smaller and the boundary shifts more closer to the fridge especially in Y-axis (Figure 7.5b) when I increase the size of the training dataset. This can be justified with the tendency in Subsection 7.8.1 to approach to the fridge from the right direction (i.e. $Y+$ direction). This over-estimation is more restricted in this subsection with positive trials whose opening position is more centric. Thus, there exists a more restricted region-of-interest in the heat map (Figure 7.5b).

7.8.3 Positive and Negative GMMs using Trials from 7 Episodes

There are tendencies to generalize the probability of opening with success when the robot is closer to the fridge in the positive-only models. However, this is an

over-generalization because it is more likely to suffer from the problem of the handle being realized by the robot arm when the robot is too close. In this subsection, I take into account 7 positive and 18 negative attempts reside in the first 7 episodes. As the negative dataset contains the example of handle being realized, the respective model suffers less from the over-generalization about proximity (Figure 7.5c).

7.8.4 Positive and Negative GMMs using Trials from 15 Episodes

The region-of-interest in the heat map becomes smaller when I include all available trials in the available NEEMs (Figure 7.5d). In this case, the model becomes more immune to over-generalization with more failed trials from different positions. Thus, the robot has a better (and more restricted) model to sample for positioning itself in the new episodes of opening fridge actions.

7.8.5 Discussion

One of the key factors towards high success rates in complex manipulation is to locate self properly in terms of the affordances such *perceiveability*, *reachability* and *usability*. For this purpose, I have presented an affordance modeling methodology based on Gaussian regression. In the experiments, I have shown that the presented methodology successfully predicts which positions can lead to success in terms of both the door handle's *reachability* and the door's *usability* (i.e. the ability to open the closed door wide enough for manipulating inside).

As most of the manipulation actions require similar correct positioning before executions, I believe that this methodology can be used for those actions as well. For instance, in a cooking task, the robot needs to correctly position itself for both placing the pot onto the heater (*reachability*) and operating the turning wheel of the heater (*usability*).

Its integration with a symbolic knowledge base, which contains a rich set of semantically-rich and generic execution logs, NEEMs, further eases the method's generalizability. For instance, in the case of the aforementioned cooking task, the high-level planner needs only to adapt the task type inside the query for obtaining parameters (The original query is given in Section 8.3.1):

```
?-findall(FeatureList,
      (entity(Act, [an, action,
```

```
      ['task_context', 'CookingPot']]),
    task_start(Act, StartTime),
    task_success(Act, true),
    belief_at(robot('base_link', RobotPose),
      StartTime),
    pose_into_relative_coord(RobotPose,
      FridgePose, RelativePose),
    matrix_translation(RelativePose, RelativeTransf),
    matrix_rotation(RelativePose, RelativeRot),
    append(RelativeTransf, RelativeRot, FeatureList)),
  FeatureListofLists),
  generate_feature_files(FeatureListofLists,
    'positive.csv').
```

Lastly, I want to emphasize that the affordance models generated in this chapter should not be considered as one of the scientific end-products in this dissertation. Instead, these models are intermediate results which will enable robots to **adapt** their actions to new conditions as described in Chapter 8.

Chapter eight

Action Adaptation

In order to perform human-level tasks flexibly in varying conditions, robots need a mechanism that allows them to exchange knowledge between themselves. One approach to achieve this is to equip a cloud application with a set of encyclopedic knowledge (i.e. ontologies) and execution logs of different robots performing the same tasks in different environments. In this chapter, I present the adaptation layer used for generalizing the actions stored inside NEEMs. This layer uses the cloud interface described in Chapter 4. In the experiments, fridge-opening actions are conducted by two PR2 robots and one Fetch robot which are located in two different kitchens.

The research and scientific results described in this chapter is also presented in (Bozcuoglu et al., 2018a).

8.1 Symbolic Knowledge stored in Ontologies

As presented in Section 2.2.1, KnowRob is used as the framework for knowledge representation and reasoning tool. It processes ontologies in W3C Web Ontology Language (OWL) (Bechhofer et al., 2004). The ontologies about the household, manipulation actions and service robots come standard with KnowRob. Apart from these standard ones, other task-specific ontologies may also be processed on-demand.

KnowRob can process semantic descriptions of environments in the format of *semantic maps*. As presented before, these maps contain knowledge about objects in the corresponding environment such as physical properties and pose information. Using them, one can also link objects to knowledge from other ontologies. For instance, if there exists *Object_A* of type *Fridge* in the semantic map of *Environment_A*, it can be inferred that *Object_A* is a container because type *Container* is defined as a superclass of type *Fridge* in the human household knowledge ontology.

By combining knowledge resides in multiple sources during reasoning, KnowRob is able to find out useful insights for robotic actions. In Figure 8.1, the robot is delegated *fetch-an-object* task for a *milk box* in the environment. The plan first queries for the superclass of type *milk box* which is answered as *perishable*. Then, the robot queries for a location that may contain *perishable* items. In the kitchen ontology, the fridge is defined as the likely location for *perishable*. Thus, it returns the fridge instance that exists in the environment.

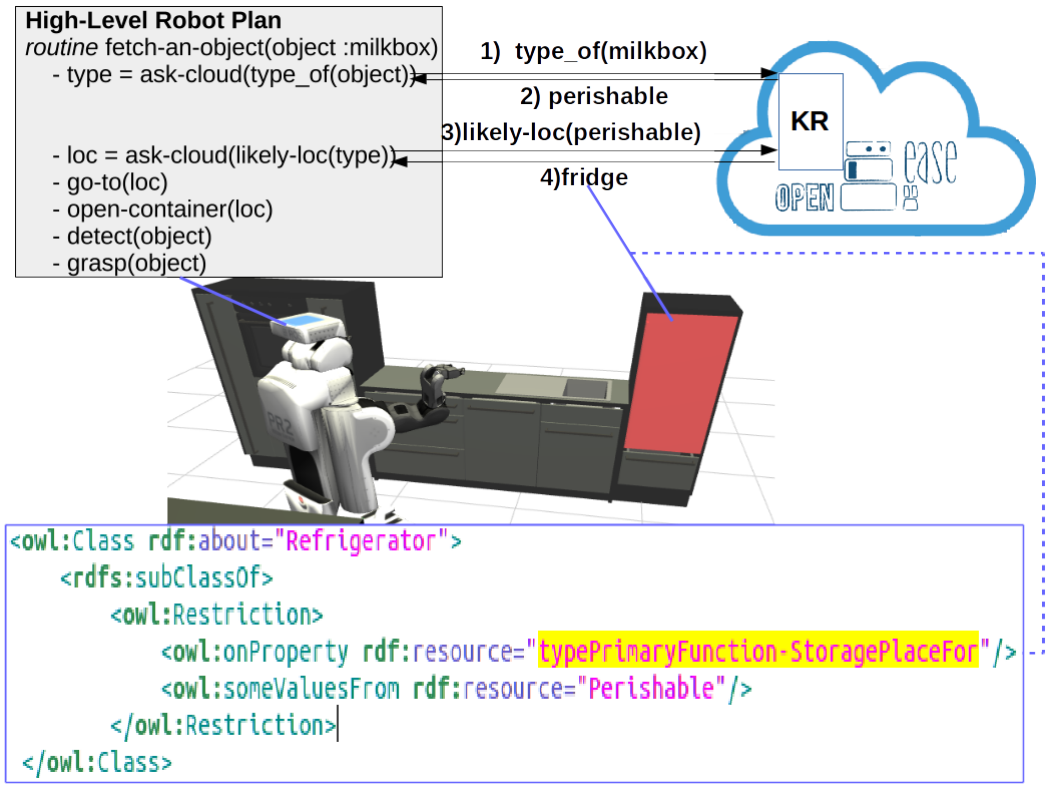


Figure 8.1: *An example of how the knowledge is supplied to high-level plans (KR is abbreviation for KnowRob).*

8.2 Reasoning using NEEMs and Ontologies

With NEEMs registered, the knowledge base has an additional source of information. As already outlined in Section 2.3, using these "past memories" together with ontologies, robots can reason about which parameters led to successful executions and which conditions resulted in failures.

For example, Figure 8.2 depicts a use-case that PR2 asks how it grasped *pancake mix* with success. The remote knowledge base, openEASE, responds with a pose of the robot along with the action parameters.

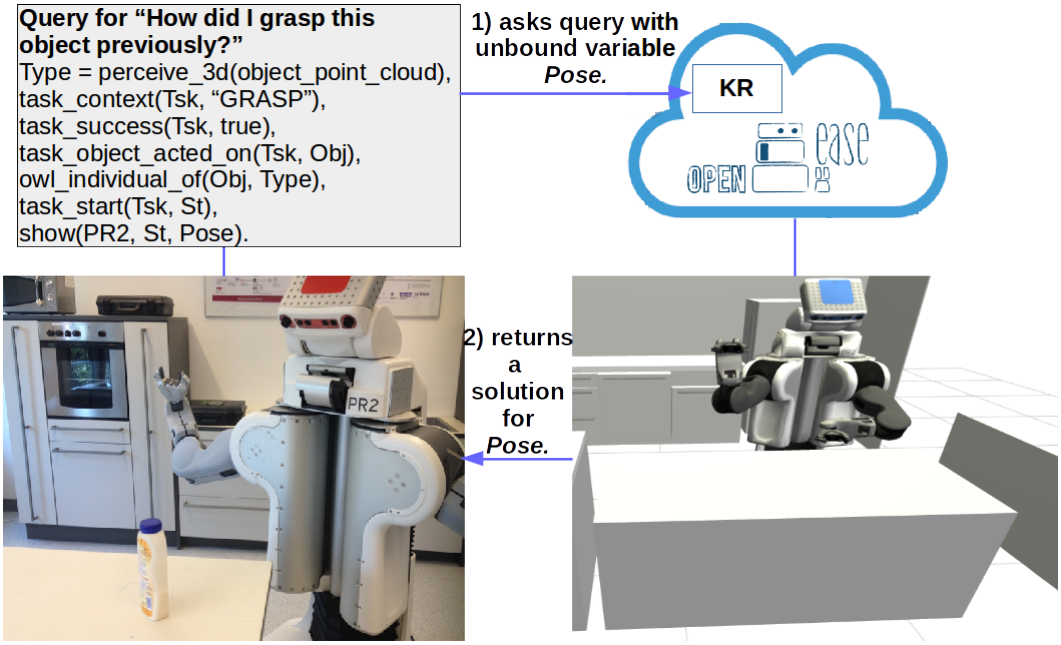


Figure 8.2: *An example use case of openEASE equipped with episodic memories.*

8.3 Action Generalization via Knowledge Transfer

In this section, I provide some multi-robot scenarios in which robots adapt others' plan parameters and subsymbolic data using openEASE.

Rule 1: knowrob:'RotateEndEffector'

Condition	Effect	Description
$(width(Handle_t) > height(Handle_t) \wedge width(Handle_s) < height(Handle_s)) \vee (width(Handle_t) < height(Handle_t) \wedge width(Handle_s) > height(Handle_s))$	Rotate end effector during handle grasping by 90°	If the relation between the handles width and height is different in the target map compared to the source map, then we have to rotate the handle grasping position. I have fixed this rule's effect to 90° since most of door handles are either parallel or perpendicular to the ground. On the other hand, this rule can be further extendable by taking into account of the desired angle effect.

Rule 2: knowrob:'ChangeHand'

Condition	Effect	Description
$(left\_of(location(Hinge_t), Door_t) \wedge right\_of(location(Hinge_s), Door_s)) \vee (right\_of(location(Hinge_t), Door_t) \wedge left\_of(location(Hinge_s), Door_s))$	Use the other hand	In order to decide whether or not the same arm should be used for opening the new container, this rule checks where the hinge is located in the container.

Rule 3: knowrob:'ChangeGraspingFace'

Condition	Effect	Description
$is\_cluttered_t(grasping\_side_s(Object))$	Grasp the object from another side	If the object cannot be grasped in the new map from the same side because of obstacles, then the robot will try another side. Although this rule does not guarantee a successful grasp. it is likely that most of the objects in household such as milk boxes can also be grasped from the opposite side with same technique and similar parameters.

Table 8.1: *Semantic rules for adapting plan parameters. The suffix $_s$ refers to the old(source) map and $_t$ refers to the new(target) map.*

Rule 1: knowrob:'ChangeRadiusOfTrajectory'

Condition	Effect	Description
$dist(Handle_t - Hinge_s) \neq dist(Handle_t - Hinge_s)$	Change radius by $dist(Handle_t - Hinge_t) - dist(Handle_s - Hinge_s)$	If there is a difference in the distance between the hinge and handle in the target and in the source, this rule adapts the arch-like trajectory centered at the hinge by changing the radius.

Rule 2: knowrob:'RotateTrajectoryAroundX'

Condition	Effect	Description
$angular\_distance(opening\_direction_t, opening\_direction_s) \neq 0^\circ$	Rotate trajectory around X-axis by $angular\_distance(opening\_direction_t, opening\_direction_s)$	Adapt the trajectory such that it creates an arch around the hinge. It is assumed that the X-axis points from the front of the container to its back and the Z-axis is perpendicular to the ground.

Table 8.2: *Semantic rules for adapting subsymbolic data. The suffix $_s$ refers to the old(source) map and $_t$ refers to the new(target) map.*

8.3.1 Adapting Plan Parameters Semantically

Physical variances might require changes in the plan parameters. When this is the case, autonomous robots should be able to find out how these changes should occur. In order to accomplish this, they must take into account the semantic maps of both the source and target environments together with the NEEMs recorded in the former. For instance, if a robot wants to adapt the parameters of a *fridge-opening* to an *oven-opening* task, it must first adapt its mobile base positioning according to the opening radius so that the oven door does not collide with the body. In addition, it must adapt its end-effector pose according to the shape of the oven handle. In other words, if the fridge handle is perpendicular to the ground but the oven handle is parallel, then the grasping pose for the handle has to be rotated by 90°.

For this purpose, I have proposed two algorithms, i.e. Algorithm 1 and Algorithm 2. The former accepts a source NEEM together with the semantic maps of both environments as input. For each task, the primary object which was interacted during the task is visited. Then, it tries to find an object which has a common superclass type in the target environment. When a match is found, it compares the properties of these objects and returns the changes in properties together with the corresponding tasks.

After these changes have been determined, Algorithm 2 adapts the parameters of each task accordingly. These tasks are, then, asserted to the KnowRob reasoning engine as *IntentionalMentalEvent*, which implies that these tasks are only conceived and not yet executed.

Algorithm 1 Changes in env. properties that matter for plan.

```
1:  procedure FINDCHANGES( EpisodicMemory, Map,
                           NewMap)
2:      TaskChanges ← []
3:      for each Task in EpisodicMemory do
4:          Obj ← objectActedOn(Task)
5:          Props_Obj ← props(Obj, Map)
6:          find a NewObj such that
                super(type(NewObj)) = super(type(Obj))
                ∧ in(NewObj, NewMap)
7:          Props_NewObj ← props(NewObj, NewMap)
8:          append<Task, adapt(Props_Obj, Props_NewObj)>
                to TaskChanges
9:      end for
10:     return TaskChanges
11: end procedure
```

Algorithm 2 Adapting parameters according to changes.

```
1:  procedure ADAPTCHANGES(TaskChanges)
2:      NewTasks ← []
3:      for each <Task, Change> in TaskChanges do
4:          params ← executed_with_parameters(Task)
5:          params_new ← apply(params, Change)
6:          Type ← type(Task)
7:          NewTask ← generate(Type, params_new)
8:          append NewTask to NewTasks
9:      end for
10:     return NewTasks
11: end procedure
```

In the line 5 of Algorithm 2, the *apply* procedure returns a new set of parameters according to the semantic rules defined in KnowRob. Although one may think that defining such rules cannot scale for all types of objects, these rules are mainly about pure geometrical properties and are used with common superclasses as shown in Algorithm 1. For example, oven, microwave, and fridge can be united as *HingedJoint-Container*. A position candidate from which the robot can easily open the container depends on the radius of the container's door and not on what type of *HingedJoint-Container* the robot is working. Some knowledge adapting rules that are currently available are given in Table 8.1.

8.3.2 Adapting Subsymbolic Data Semantically

Similar to the adaptation of plan parameters (Section 8.3.1), subsymbolic data can be adapted to a target environment by comparing geometrical properties in the source and target objects. For instance, an opening-a-fridge trajectory can be adapted to an oven-opening task as both tasks require arch-like trajectories centered at a hinge. To adapt one to another, the dimensions of the doors (for changing the radius), and the hinge location w.r.t. the doors (for rotating the trajectory samples) need to be considered.

For this purpose, I have slightly modified Algorithm 1 and Algorithm 2 since the implementation in a first-order logic programming language, like SWI-Prolog (Wielemaker, 2003), requires some changes in the algorithmic steps which do not affect the end result. Upon this, I defined the semantic rules in Table 8.2. Similar to trajectories, affordance models described in Chapter 7 can be adapted to different physical properties using these rules.

8.4 Experiments

In this section, I report how we (Bozcuoglu et al., 2018a) have tested this methodology. The experimental setup involved one PR2 robot (a.k.a. JSK-PR2) and one Fetch robot, both located in the kitchen environment of JSK Lab, and an additional PR2 robot (a.k.a. Raphael) operated in IAI Lab kitchen. We have recorded episodic memories to openEASE while the JSK-PR2 executes *bring-ingredients-for-conflakes* plan (Figure 8.3). In this plan, the robot first opens up the fridge for fetching a milk box. After it puts the milk box on the table, it brings a bowl and a cornflakes box from the kitchen counter.

During these executions, JSK-PR2 approaches to the fridge randomly within a predefined upper-limit distance and tries to open it. By randomizing locations, we aim to have an affordance model for *fridge-opening* in the means of unimodal Gaussian distributions. After 58 executions which include 27 successful and 31 failed *fridge-opening* trials, the affordance model generated by openEASE enables JSK-PR2 to approach and open the fridge door with 100% success in 20 test trials. Thus, we have used this model in the IAI Lab kitchen as described in Section 8.4.2.

The semantic maps of both kitchens are available in openEASE such that the agents can infer the differences in the maps and adapt the plans according to the rules

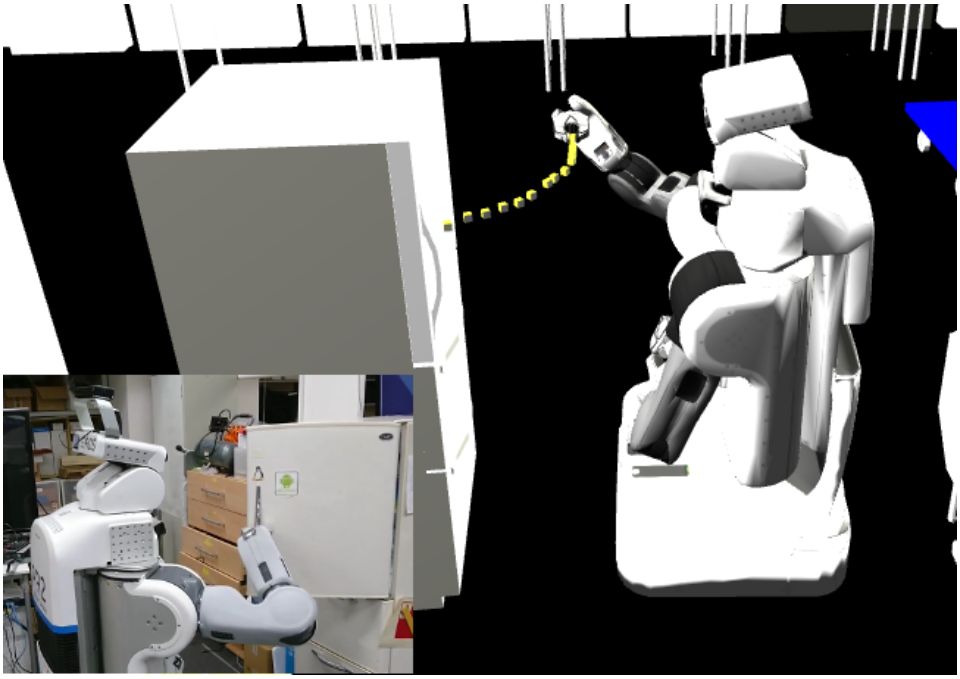


Figure 8.3: *The trajectory of fridge-opening from openEASE as recorded to the corresponding episodic memory.*

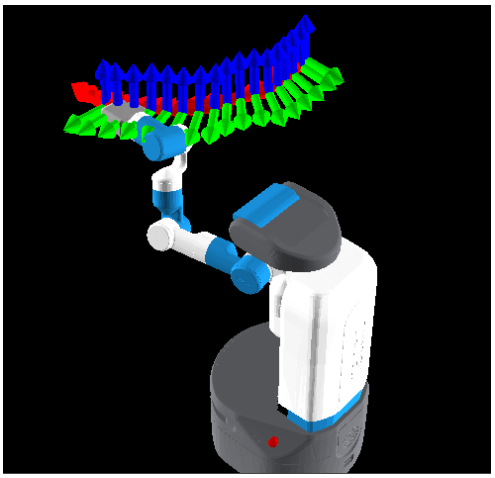
mentioned in Table 8.1 and 8.2.

8.4.1 Use Case 1: Adapting Opening Action to a Different Robot Platform

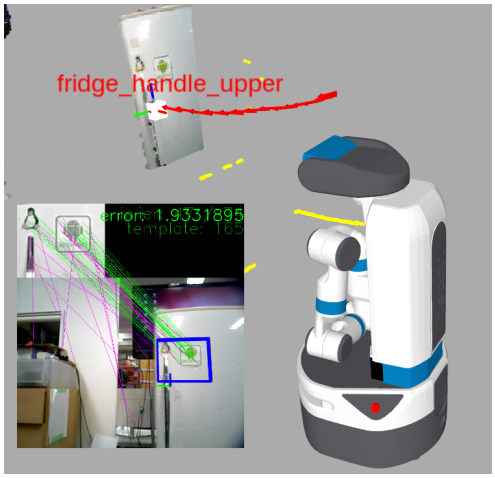
In this use case, the aim is to adapt the actions recorded in a NEEM to a different type of robot. As NEEMs include trajectories of opening the fridge door recorded using JSK-PR2, we (Bozcuoglu et al., 2018a) confirmed that a single-armed robot, Fetch, can also open the fridge in the same environment.

Each robot has a unique kinematic model and a different configuration of actuators-sensors, which make plug-and-play joint-level data such as trajectories to the target robot impossible. Because of this, this methodology first converts the trajectory representation from joint-level data to an end-effector based representation that uses global coordinates (Figure 8.4a).

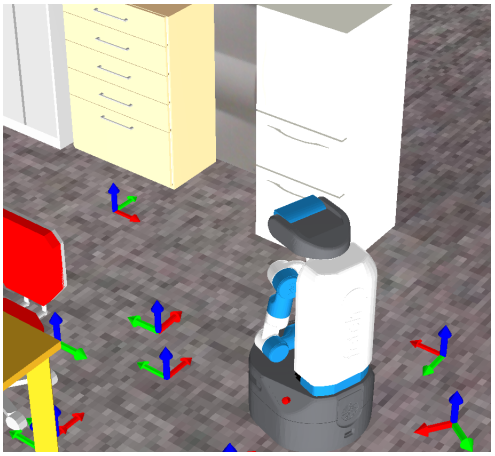
Adapting actions based on such representations is highly dependent on the precision of robot localization. In order to make it more stable, we (Bozcuoglu et al., 2018a) have transformed these trajectories by fitting the first trajectory point to the handle pose detected with perception (Figure 8.4b).



(a) End-effector-level trajectory data from episodic memory



(b) Fitting trajectory with perceived fridge handle



(c) Environment model with "spots"

(d) Fetch robot opening fridge door using transferred trajectory

Figure 8.4: *Fetch robot using knowledge generated by PR2*

For the navigation, both JSK-PR2 and the Fetch robot have shared geometric annotations which are semantically grounded to locations in the semantic which is referred to as "spots" (Figure 8.4c).

We (Bozcuoglu et al., 2018a) have demonstrated that the target robot successfully adapt the door-opening task using transferred knowledge from JSK-PR2 (Figure 8.4d).

8.4.2 Use Case 2: Adapting Opening Action to a Different Environment

Raphael adapted an opening-fridge task for its own setting in the IAI Lab kitchen from a NEEM which JSK-PR2 recorded during opening a fridge in the JSK kitchen.

The CRAM high-level plan for this adaptation and execution of a recorded task consisted of two main subtasks: (1) traversing to a pose where Raphael can open the fridge, i.e. the trajectory is reachable and has no collisions with the environment happen, and (2) executing the trajectory of opening a fridge inside the corresponding NEEM (either the original or adapted version). In addition to the parameters of the robot platform, as discussed in Section 8.4.1, the standing point of the robot also depends on the arm used (e.g., left or right), the physical properties of the manipulated object, the environment etc. Thus, these parameters are expected to differ from JSK Lab. to IAI's kitchen, therefore, the robot base pose that a PR2 used in the JSK kitchen often cannot be used as is in the IAI kitchen.

Instead, we (Bozcuoglu et al., 2018a) adapt the affordance model generated for fridge opening in JSK Lab with the mechanism described in Section 8.3.2. JSK-PR2 have performed the task 58 times, from which 27 were successful. Each time the robot was positioned slightly differently with respect to the fridge, using a random offset on the navigation pose. To find the region of poses with a high probability of success as an answer to the robot's query, openEASE fits a unimodal Gaussian distribution to all the successful and unsuccessful trials of fridge opening task it can find in NEEMs as described in Chapter 7. Thereafter, the reference object for the pose distribution for opening a *HingedJointContainer* is taken as its handle. As coordinate systems of objects in the semantic map are defined by its designer, one should expect differences during the adaptation. Thus, an additional coordinate system is defined based on the relative positioning of the handle and the fridge door joint. Then, all the relevant geometric data from both kitchen environments is transformed using that coordinate frame. The mean and the covariance of the learned Gaussian are, thus, transferred into Raphael's environment, which results in a distribution visualized in Figure 8.5a.

Once Raphael locates itself appropriately, it asks the queries, "*Which arm should be used in the fridge opening task*" and "*What is the trajectory of the gripper?*". The trajectory cannot be used as is and has to be adapted to the new environment. For this, openEASE has applied the rules described in 8.3.2. Figure 8.5b depicts the original trajectory as is (in purple with a smaller radius) and the adapted version (in red with a bigger radius).

In this use case, NEEMs are used as a bias for performing a similar task in a different

environment. Using the affordance modeling, all poses where the robot can stand are constrained to a Gaussian distribution but the poses where the probability is higher than 0 are still not guaranteed to succeed and a possibility of failure has to be considered. For example, the dimensions of the fridge door might be much bigger such that not all the trajectory points are reachable anymore, or there might be an obstruction in the environment which makes navigation to a particular pose impossible. Thus, we describe the location to stand to successfully execute a trajectory using symbolic descriptions in order to define such constraints.

The plan is executed in the fast plan projection environment (Mösenlechner and Beetz, 2013) (Fig. 8.5c) to test different samples from the distribution. The one that does not throw a reachability or environment collision failure is chosen for the execution on the real robot (Fig. 8.5d).

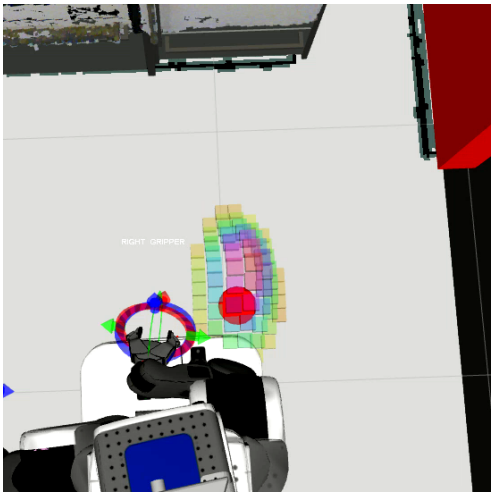
8.5 Analysis of Experimental Results

This chapter presents how **adaptation layer** enables robots to connect and make use of the data generated even by another type of robot with different kinematics and control architecture. In the experiments, I have reported the adaptability of opening trajectories to different environments using episodic memories generated in JSK Lab. Such an ability is very valuable for accessing big data and remote knowledge processing facilities. On the other hand, this is not enough to solve the problem alone. Robots themselves should also have the necessary capability to identify which portions of data they can use and how they can do so. For instance, it is important for Fetch robot to check if it can reuse the trajectory that JSK-PR2 has generated before the execution.

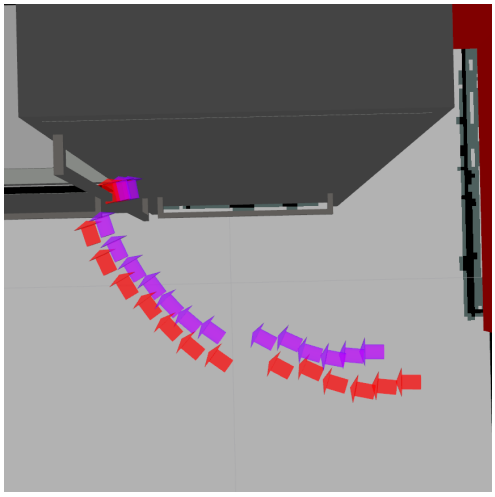
Section 8.4.2 illustrates how an agent can adapt the data from a source environment to target environment by using available rules and semantic descriptions of both environments. I believe that this is an important step towards robots mastering a particular task in a sense that it can perform this task under a variety of conditions and in different environments. At the same time, I also consider that generating semantic descriptions of the environments still requires a lot of human input and manual processing. To automate this process, roboticists need to improve perception such that robots can recognize the types of objects with their distinctive features while they are navigating in an unknown environment. Finally, roboticists need to ensure scalability of symbolic rules to all of service robotics tasks by applying the

presented methodology to other tasks.

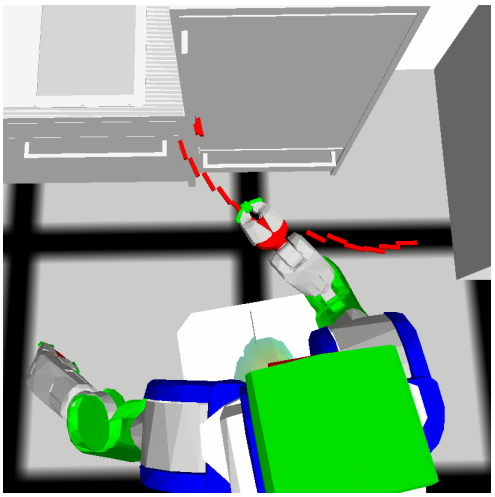
I foresee that the proposed methodology can, also, be generalized to other cloud platforms, robots, and actions. Having semantic descriptions of robots and environments is the only prerequisite for this manner of approach. Although one might think an extension of the rule set is essential for this, I believe that a fairly small set of rules will suffice. Since these rules are mainly about pure geometrical properties, very different actions can also make use of the same rule to be adapted in target environments. In cases that these rules do not suffice, I believe that augmenting symbolic rules with machine learning approaches can offer us some hybrid mechanisms to deal with the complexity of such adaptations.



(a) Pose distribution for the robot's base to successfuly execute a task



(b) Trajectory adapted to the new environment (red) and the original one (purple)



(c) Performing the task in fast plan projection mode

(d) PR2 robot opening the fridge using the transferred trajectory

Figure 8.5: *PR2 robot performing the fridge opening task in a new environment*

Part IV

Experience Verification

Chapter nine

Consistency Checking in Experimental Knowledge

In this chapter, a methodology for rendering semantically-annotated videos out of NEEMs is presented. This methodology does not only prove that NEEM is a well-detailed episodic memory model, which is adequate for reconstructing world state at any time during the execution, but also it can be used for analyzing, diagnosing and debugging robotic experiments I mainly use this methodology throughout my doctoral studies for checking the inconsistencies between the belief state and sensory data in data before actually making use.

The research and scientific results in this chapter are also presented in (Bozcuoglu et al., 2015a).

This chapter is organized as follows: First, I describe the proposed methodology in detail (Section 9.2). Then, the predicates used inside KnowRob for interfacing with this module is given in Section 9.3. Finally, I illustrate some examples of how it works and analyze how this methodology can be helpful.

9.1 Rendering Videos out of Episodic Memories

The video generation methodology is implemented as a part of openEASE and can be interfaced via openEASE web interface. After logging in, openEASE users can select *Episodic Memory Replay* from menu. Then, users can choose an existing *video setup* or create their video setup. A video setup consists of a start and end time, an initial query for loading the corresponding episodic memory, semantic map, and robot description, and, finally, an animation query.

After the user specifies the setup, the rendering process can start. Firstly, *Initial Query* is executed to visualize the semantic map in 3D canvas and to load the log file of the experiment run in the knowledge base. After that, *Animation Query* has a hidden time variable T, which has a value range between start and end time, and this variable is increased by $1/FPS$ after completion of each iteration (Algorithm 3). Thus, it is executed for each time step to generate frames of the video.

Algorithm 3 Video rendering algorithm

```
1: procedure VIDEO–RENDERING
2:     Execute Initial Query
3:     t_start = start time of the experiment
4:     t_end = end time of the experiment
5:     while t_start < T < t_end do
6:         Execute Animation Query with T
7:         Update T as T = T + 1/FPS
8:     end while
9: end procedure
```

9.2 Extracting World State from Episodic Memories

When the user initiates the rendering process, the web interface starts to send queries to the knowledge base. In return, it has the world model at the current time step by checking episodic memories, the semantic map, and loaded ontologies. The corresponding model is rendered in the 3D canvas using openEASE visualization module. By taking snapshots of this canvas for every time-step, the system records frames of the desired video. This methodology is depicted in Figure 9.1.

The video resolution is depending on how precise the environment and the robot

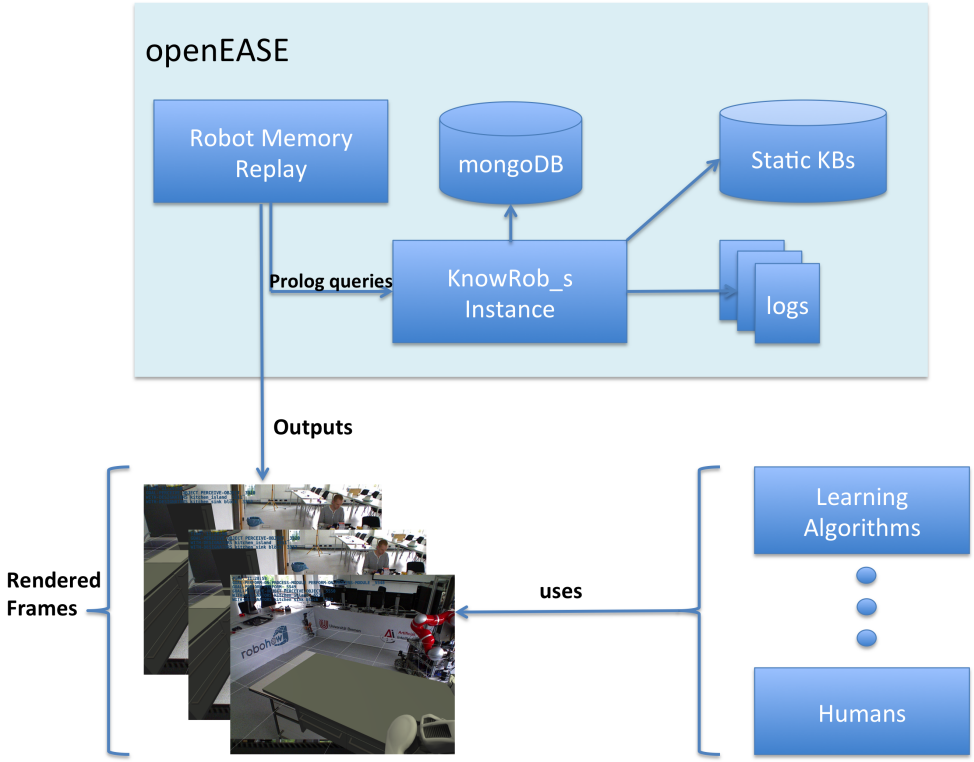


Figure 9.1: *Proposed methodology for rendering videos out of NEEMs.*

model are represented in the belief state as well as the accuracy of the perception.

9.3 Rendering Videos out of NEEMs

openEASE users can write video queries by using standard OWL/RDF library and KnowRob predicates that are defined by Winkler et al. (2014) and Beetz et al. (2015) as well as the dedicated predicates specialized for rendering of videos (Table 9.1). These predicates are divided into three groups. In the first group, there are the utility predicates for asserting semantic annotations on video frames such as speech bubbles and hud text. Secondly, there exists a group of predicates which are for inferring the robot's goal, task context and belief state at a given time, t. The last group of predicates is for listing and playing already rendered videos.

Predicates for the annotations of video	
camera_pose(Position, Orientation)	Sets the camera pose to the given position and prientation
show_speech_between_interval(Text, Position, T_{Start}, T_{End}, $T_{Current}$)	Displays given text as a speech bubble at Position if $T_{Start} < T_{Current} < T_{End}$
show_speech_after_time(Text, Position, T_{After}, $T_{Current}$)	Displays given text as a speech bubble at Position if $T_{Current} > T_{After}$
show_speech_before_time(Text, Position, T_{Before}, $T_{Current}$)	Displays given text as a speech bubble at Position if $T_{Current} < T_{Before}$
show_hud_text_between_interval(Text, T_{Start}, T_{End}, $T_{Current}$)	Displays given text as a hud text if $T_{Start} < T_{Current} < T_{End}$
show_hud_text_after_time(Text, T_{After}, $T_{Current}$)	Displays given text as a hud text if $T_{Current} > T_{After}$
show_hud_text_before_time(Text, T_{Before}, $T_{Current}$)	Displays given text as a hud text if $T_{Current} < T_{Before}$
Predicates for acquiring the robot's intentions at a certain time	
get_goals_at_time(T, Goals)	Returns the active goals of the robot at time *T*.
get_contexts_at_time(T, Contexts)	Returns the contexts of the tasks that are active at time *T*.
get_perception_at_time(T, Perception)	Returns the latest perception designators at time *T*.
get_active_designators_at_time(T, Fields, Props, Results)	Returns the given fields of active designators at time *T*.
Predicates for already rendered videos in the server	
experiment_videos(ExpName, URLs)	Returns urls of the rendered videos given the experiment name.
video_play(URL)	Plays the video at the given url inside the image canvas.

Table 9.1: *Video rendering predicates.*

9.4 Use Case 1: Rendering a Video Annotated With Active Goals

In this case, PR2 performs grasping to a spatula during a *pick and place* plan in a kitchen environment. *Initial Query* loads the episodic memory, the robot model and the semantic map of the kitchen, and visualizes the initial world state in the canvas. In addition to the world state, the trajectory of the left arm during the task execution is also visualized:

```
?-load_experiment('pick-and-place/log.owl'),
  owl_parse('kr://robot_description/PR2.owl'),
  owl_parse('kr://semantic_maps/room.owl'),
  add_object_with_children('http://knowrob.org
  /ias_semantic_map.owl#SemanticMapPM580j'),
```

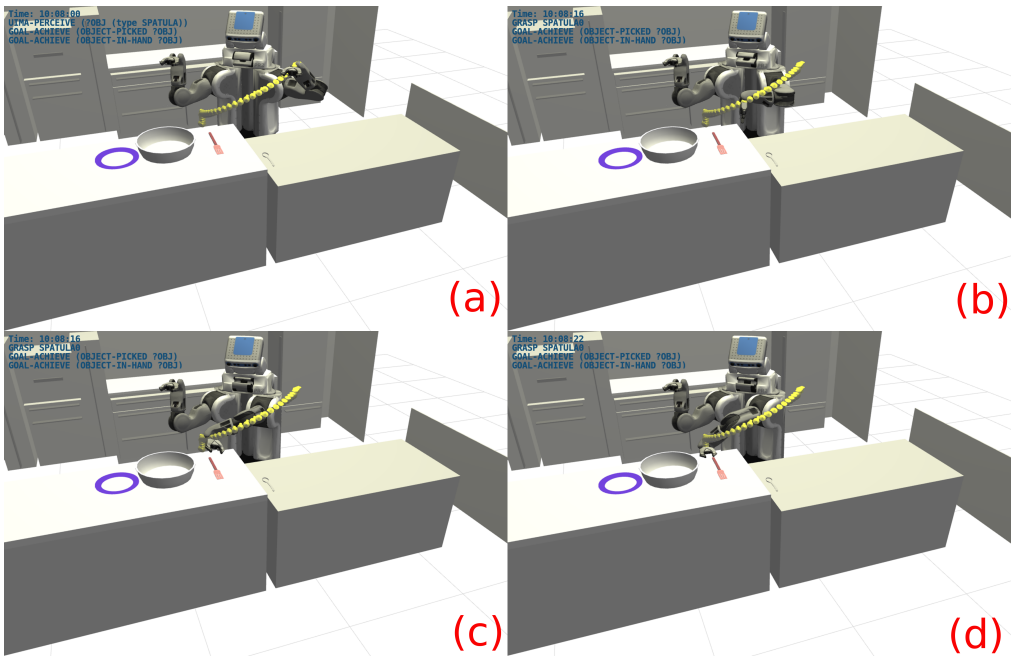


Figure 9.2: *Frames of the video generated in Section 9.4.*

```
add_trajectory('l_wrist_roll_link',
T_st, T_end, 0.1).
```

For each time step, the robot position is updated accordingly. Besides, the video is annotated with the active goals of the robot as a HUD text along with the experiment time and the perception details while a perception task is taking place. Thus, *Animation Query* is as follows:

```
?-add_agent_visualization(pr2:'PR21',T),
  get_goals_at_time(T, Goals),
  show_hud_text_between_interval
  (Goals, T_st , T_end , T).
```

Some frames of the resulting video are shown in Figure 9.2.

9.5 Use Case 2: Rendering a Video from Robot's Eye Perspective

In this use case, the previous example is extended with two features (Figure 9.3). First, the camera pose of the video is updated to the pose of PR2's HD camera frame in every time step. Second, the speech bubbles are used for the annotation of objects with perception data.

```
?-add_agent_visualization(pr2:'PR21',T),
  get_perception_at_time(T, Perc),
  show_speech_between_interval
  (Perc, Pos(Perc), T_st , T_end , T),
  mng_lookup_transform('/map',
  '/high_def_optical_frame', T, P),
  camera_pose(Pos(P), Orientation(P)).
```

The predicate, *show_speech_between_interval*, is used for annotating the perceived object properties as a speech bubble. For updating the camera pose, the transformation matrix of PR2's camera from the reference frame is acquired and, then, it is passed to the *camera_pose* predicate.

9.6 Use Case 3: Belief-State Consistency

By observing current actions, humans can estimate possible actions in the future based on their expectations (Hegarty, 2004). An example would be classifying physical structures as stable or unstable based on expectations of the structure physics (e.g., the centroid of structure parts) and the material of the structure (e.g., the weight of the material). They can identify situations where it is evident that the video does not represent the reality accurately watching a video generated from episodic memory of a robot. For example, using common sense, an unexpected or sudden movement of body parts that does not make sense in the context of the currently executed task or that is physically impossible can be identified easily by humans. Such inconsistencies can be due to bugs in inference mechanisms or control programs or inaccuracy of sensory data or belief state (e.g., robot localization error). Thus, rendered videos from episodic memories are an effective way to identify inconsistencies in NEEMs by allowing humans to visually and efficiently inspect the knowledge base based on their expectations.

Being able to observe the actual execution from a camera stream influences human expectations. Instead of estimating the next state based on physics, users know the next state from watching the episode in reality. Robot control programs can record the episode execution using a camera sensor attached to the robot. Furthermore, using the aforementioned videos, it is possible to generate a video based on knowledge about the episode using the same camera configuration that was used while recording the episode in reality. In other words, one can see where the episodic knowledge does not match with the reality by rendering the generated video on top of the video that was recorded by the robot camera. Thus, it can be visually examined if the episode

knowledge is consistent with the reality by watching such an augmented reality video of the episode. An example can be seen in Figure 9.4 where PR2 is operating in a kitchen environment manipulating with various objects. The animation query that generates these frames is as follows:

```
?-add_agent_visualization(pr2:'PR21',T),
  get_perception_at_time(T, Perc),
  mng_lookup_transform('/map',
  '/high_def_optical_frame', T, P),
  camera_pose(Pos(P), Orientation(P)),
  mng_query('image', one(DBObj), 'header.stamp', T,1),
  marker(background_image('CurrentImg'), Marker).
```

where the camera image for time, t, is retrieved from mongoDB with *mng_query* predicate and, then, is put as a background image of the canvas with *marker* predicate.

At first, there exists a significant difference between the robot's belief state and image stream from the camera where the furniture is located is not aligned with the images taken with the robot's RGB-D camera. In contrary, in the last frame, the difference is negligible.

9.7 Discussion

Creating videos out of NEEMs can be valuable in various ways. First, robots create logs that contain gigabytes of data during execution of a relatively long experiment. By watching these videos, one can find out inconsistencies between the sensory data and the belief state without checking such enormous size of data. Secondly, these videos can be seen execution summaries with annotations which reflect what robots think, plan, reason and do. These leads help improve robot plans. Finally, these videos also prove proves that the NEEM logging infrastructure creates a comprehensive episodic memory for the robot experiments.

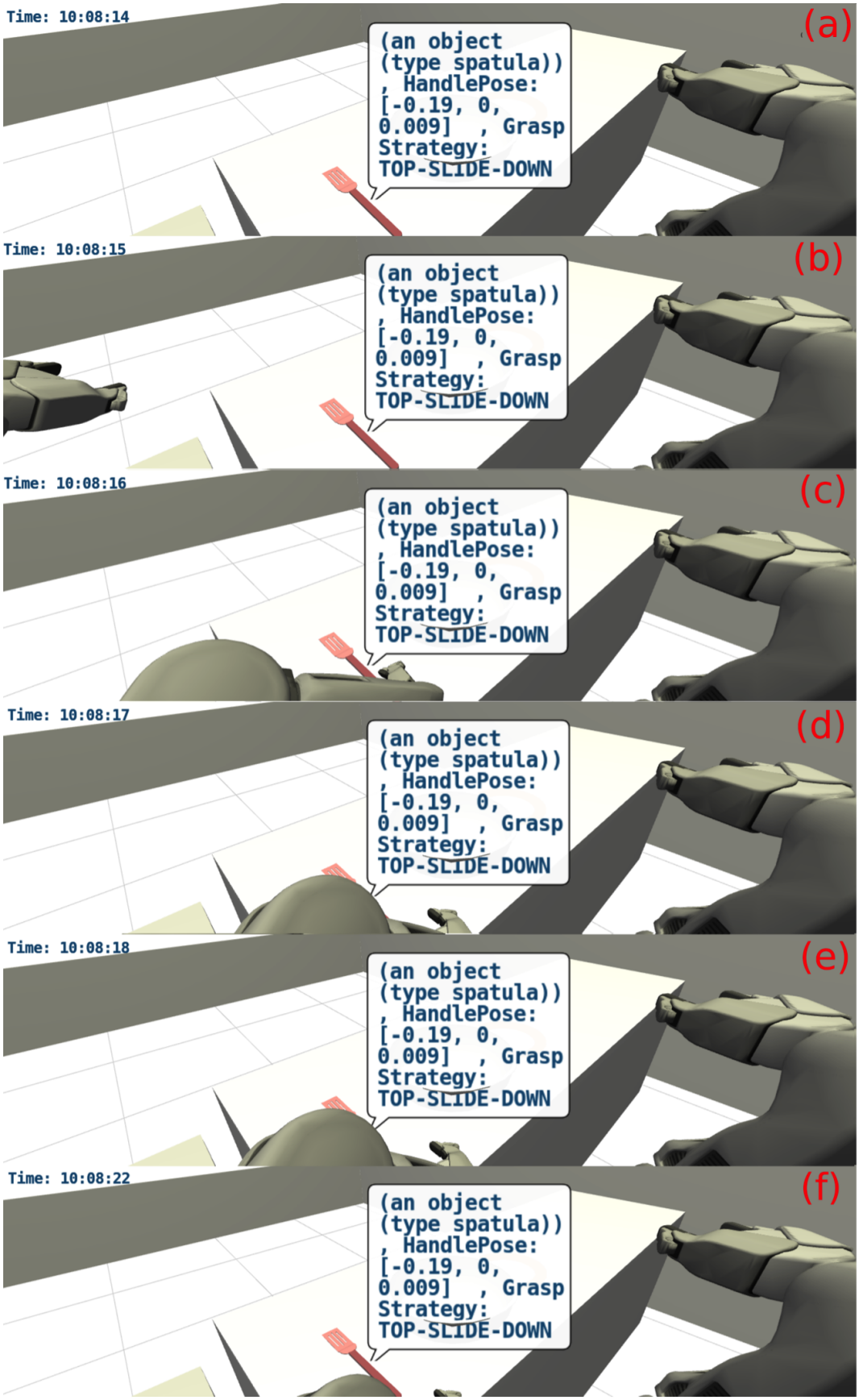


Figure 9.3: *Frames of the video generated in Section 9.5.*

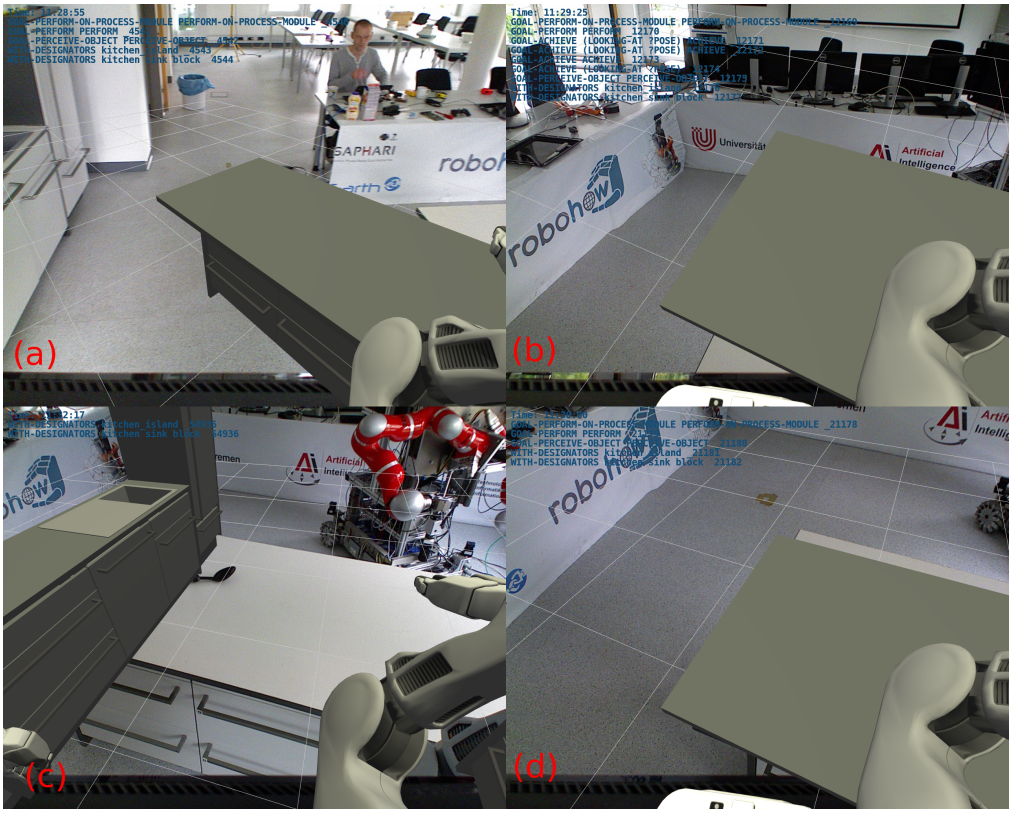


Figure 9.4: *Visual inconsistencies. In* ***(a)***, ***(b)***, ***(c)***, *there exists inconsistencies described in Section 9.6. On the other hand, in* ***(d)***, *the robot belief state and the image are consistent.*

Part V

Final Remarks and Appendix

Chapter ten

Conclusions

This dissertation has presented a fast-learning framework that enables service robots to **imitate** actions, to **infer** action parameters using **prospection**, and to **adapt** them to new situations. Humans intensely use these mechanisms to gain manipulation skills starting from infancy period. Throughout my dissertation, I have argued that these skills resemble fundamental bases for service robots to succeed in their tasks in unstructured environments such as households (more specifically, **Limitations 1&2**). With the variety of experiments distributed over the chapters, I have focussed on scientifically grounding my arguments.

In Chapter 2, I have first presented the building blocks of the presented framework along with the related concepts, formats and tools such as *narrative-enabled episodic memories* (Winkler et al., 2014) (in short NEEMs), KnowRob (Tenorth and Beetz, 2013; Beetz et al., 2018) and openEASE (Beetz et al., 2015). NEEMs store relevant information about previous robot executions and virtual reality demonstrations. KnowRob is a leading-edge knowledge representation and reasoning toolbox which allows roboticists to develop modules that feed necessary knowledge for their AI-enabled control and planning executives. openEASE acts as a remote service which enables robots and roboticists to interact with each other's knowledge base and NEEMs with ease.

After giving a comprehensive literature survey in Chapter 3, I have focused on using NEEMs available in the cloud platform, openEASE. For this manner, I have first described how roboticists can implement their cloud interface in Chapter 4. Moreover, I described a methodology for modeling affordances for the actions recorded in

NEEMs with the desired features.

Chapter 5 explains the imitation learning methodology integrated into the framework. In the meta-learning process, a meta-model is trained using many human demonstrations in virtual reality and one corresponding execution of the training tasks. This model is, later, used for the fast imitation of new tasks. In a way, such an approach is an important "learning to learn" capability, which enables service robots to learn new manipulation tasks quickly.

The provided experiments in Chapter 5 have proven that robots can infer what the task goal is and reach that goal in a real-world setting. I anticipate that the proposed methodology can be generalized to other robots, and actions thanks to meta-learning. Having a single execution of the task by the real robot is the only prerequisite for this type of approach. With such an approach, it is also possible to crowdsource training data for the data-hungry deep-learning applications. Although one might argue that the need for robot demonstrations damages the purpose of crowdsourcing, these demonstrations are a small proportion (it was between 4% and 8% in the experiments as presented in Section 5.6.) of the overall training dataset. Thus, crowdsourcing is indeed achievable for the big percentage of data needed.

In Chapter 6, I have presented the prospection service using which robots can spawn their world and conduct reinforcement learning trials to infer correct parametrizations for their motions. By recording NEEMs, there is also a possibility to reason about the simulations. Taking the simulated trials into account, the agent can reason about in *which* trial it reached the goal in a shorter period and *what* was the shortest trajectory among these trials.

Such a reasoning capability is useful for intelligent robots to identify failures and false-positives in the learning process. In particular, for reinforcement learning, it is not always the case that the generated trajectory is better than the previous ones. Using the additional parameters that a simulated environment can supply, the robot can assess this kind of issues and make better use of learning.

Thanks to its integration with openEASE, robots can access a big NEEM set and model their affordances based on action instances in that set. In Chapter 7's experiments, I have shown that a PR2 robot could model its reachability affordance using this methodology. As most of the manipulation tasks require correct positioning similar to the reachability one presented before execution, I believe that this methodology can be used for those tasks as well. For instance, during a cooking task, the robot needs to also correctly position itself for both placing the pot onto the heater (reachability) and operating the turn wheel of the heater (usability). Its integration with KnowRob, which contains a large set of semantically-rich and generic execution logs, further

improves the method's generalizability.

In Chapter 8, the action adaptation capability of the framework is introduced. This capability can be taken as a mechanism that allows robots to exchange their experiences and adapt them into their settings by leveraging symbolic knowledge about actions, agents, and environments and reasoning. By addressing two different use cases, I have shown that two PR2 robots and one Fetch robot can successfully adapt each other's plan parameters and subsymbolic data to the experiments that they are conducting.

Finally, Part IV describes how researchers can render videos from NEEMs to analyze, diagnose and debug agents' behaviors under certain conditions or whether episodic memories are intact. This methodology also proves that NEEM is a well-detailed episodic memory model.

As also noted in the respective chapters, the work presented in this dissertation is partly available in some prior publications which are listed in Appendix A.

To sum up, I have provided an in-depth analysis of the presented learning framework together with a set of experiments that validates the scientific contribution. I believe that adapting the abilities available in this framework represent a solid base for the democratization of companion robots by increasing the manipulation abilities of robots in unstructured environments.

As the last words, roboticists should investigate human development and implement similar mechanisms for establishing similar advanced development in intelligent robots. I believe that my doctoral studies pave the way for the further development of these aforementioned mechanisms. Among other aspects, richening the imitation learning abilities with more symbolic and geometric reasoning can lead to the imitation of more complex actions from virtual reality demonstrations.

Appendix A

Prior Publications and Academic Events

At the last stages of my doctoral studies, I have initiated and led the organization of a workshop entitled "Latest Advances in Big Activity Data Sources for Robotics and New Challenges" which has taken place on October 1st, 2018 in Madrid. The workshop has been held as a part of one of the flagship conferences of IEEE Robotics and Automation Society, namely 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. My aim for organizing this workshop is to increase the impact of my research by building a community around it. The detailed information and objectives of this workshop can be found in `http://www.open-ease.org/activity-data-workshop/`.

The research reported in this dissertation also featured in the scientific papers/articles/chapters which have been published in different international conferences, journals, and books. The parts of this work drawing on content from prior publications referenced the prior works where appropriate. For the sake of completeness, this section depicts a list of these publications. One of the conference papers that is listed as [4] has been finalist for the Best Cognitive Robotics Paper Award, finalist for the Best Service Robotics Paper Award in 2018 IEEE International Conference on Robotics and Automation (the other flagship conference of IEEE Robotics and Automation Society).

Book Chapters

1. D. Beßler, A. K. Bozcuoglu, and M. Beetz. Information System for Storage, Management and Usage for Embodied Intelligent Agents. In R. Drechsler and C. Große, editors, *Information Storage from a multi-disciplinary perspective*. Springer, 2019. in press

Journal Manuscripts

1. A. K. Bozcuoglu, S. Jung, J. Kim, A. Heidu, B.-T. Zhang, and M. Beetz. Crowdsourcing big data using virtual reality and cloud roboticss. *Robotics and Automation Letters*, 2019b. submitted

2. F. Yazdani, S. Blumenthal, N. Huebel, A. K. Bozcuoglu, M. Beetz, and H. Bruyninckx. Query-based integration of heterogeneous knowledge bases for search and rescue tasks. *Robotics and Autonomous Systems (RAS)*, Special Issue on Semantic Policy and Action Representations for Autonomous Robots, 2019

3. A. K. Bozcuoglu, G. Kazhoyan, Y. Furuta, S. Stelter, M. Beetz, K. Okada, and M. Inaba. The exchange of knowledge using cloud robotics. *Robotics and Automation Letters*, 3(2):1072–1079, April 2018a. doi: 10.1109/LRA.2018.2794626

4. J. Winkler, M. Tenorth, A. K. Bozcuoglu, and M. Beetz. CRAMm – memories for robots performing everyday manipulation activities. *Advances in Cognitive Systems*, 3:47–66, 2014

Conference Papers

1. A. K. Bozcuoglu, Y. Furuta, K. Okada, M. Beetz, and M. Inaba. Continuous modeling of affordances in a symbolic knowledge base. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Montreal, Canada, 2019a. IEEE. Submitted

2. A. Haidu, D. Beßler, A. K. Bozcuoglu, and M. Beetz. Knowrob-sim: Game engine-enabled knowledge processing for cognition-enabled robot control. In *International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018. IEEE

3. F. Yazdani, G. Kazhoyan, A. K. Bozcuoglu, A. Haidu, F. Balint-Benczedi, D. Beßler, M. Pomarlan, and M. Beetz. Cognition-enabled framework for mixed human-robot rescue team. In *International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018. IEEE

4. A. K. Bozcuoglu, G. Kazhoyan, Y. Furuta, S. Stelter, M. Beetz, K. Okada, and M. Inaba. The exchange of knowledge using cloud robotics. In *International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018b. IEEE. Finalist for the Best Cognitive Robotics Paper Award, Finalist for the Best Service Robotics Paper Award

5. M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. K. Bozcuoglu, and G. Bartels. Knowrob 2.0 – a 2nd generation knowledge processing framework for cognition-enabled robotic agents. In *International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018. IEEE

6. A. K. Bozcuoglu and M. Beetz. A cloud service for robotic mental simulations. In *International Conference on Robotics and Automation (ICRA)*, Singapore, 2017. IEEE

7. J. Winkler, A. K. Bozcuoglu, M. Pomarlan, and M. Beetz. Task parametrization through multi-modal analysis of robot experiences (extended abstract). In *Proceedings of the 2017 International Conference on Autonomous Agents*, AAMAS '17, 2017

8. J. Bateman, M. Beetz, D. Beßler, A. K. Bozcuoglu, and M. Pomarlan. Heterogeneous ontologies and hybrid reasoning for service robotics: The ease framework. In *Third Iberian Robotics Conference*, ROBOT '17, Sevilla, Spain, 2017

9. M. Beetz, D. Beßler, J. Winkler, J.-H. Worch, F. Balint-Benczedi, G. Bartels, A. Billard, A. K. Bozcuoglu, Z. Fang, N. Figueroa, A. Haidu, H. Langer, A. Maldonado, A.-L. Pais, M. Tenorth, and T. Wiedemeyer. Open robotics research using web-based knowledge services. In *International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016. IEEE

10. A. K. Bozcuoglu, D. Beßler, and M. Beetz. Rendering semantically-annotated experiment videos out of robot memories. In *Proceedings of the IEEE-RAS*

International Conference on Humanoid Robots, Seoul, Korea, November 2–5 2015a

Other Publications

1. A. K. Bozcuoglu, F. Yazdani, D. Beßler, B. Togorean, and M. Beetz. Reasoning on communication between agents in a human-robot rescue team. In *Towards Intelligent Social Robots – Current Advances in Cognitive Robotics*, Seoul, Korea, 2015b

2. H. Messerschmidt, J. Winkler, A. K. Bozcuoglu, and M. Beetz. Experience based task and environment specification for plan-based robotic agents. In *27. Workshop on Planen, Scheduling und Konfigurieren, Entwerfen*, Koblenz, Germany, September 16 2013

Bibliography

R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila. Task planning for human-robot interaction. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pages 81–85. ACM, 2005.

J. R. Anderson and M. Matessa. A production system theory of serial memory. *Psychological Review*, 104:728–748, 1997.

J. R. Anderson and L. M. Reder. The fan effect: New results and new theories. *Journal of Experimental Psychology: General*, 128:186–197, 1999.

J. R. Anderson and L. J. Schooler. Reflections of the environment in memory. *Psychological science*, 2(6):396–408, 1991.

J. R. Anderson, J. G. Greeno, L. M. Reder, and H. A. Simon. Perspectives on learning, thinking, and activity. *Educational Researcher*, 29(4):11–13, 2000.

J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychological Review*, 111:1036–1060, 2004a.

J. R. Anderson, M. Myowa-Yamakoshi, and T. Matsuzawa. Contagious yawning in chimpanzees. *Proceedings of the Royal Society of London B: Biological Sciences*, 271 (Suppl 6):S468–S470, 2004b.

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, W. Zaremba, O. P. Abbeel, and W. Zaremba. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems 30*, pages 5048–5058, jul 2017.

M. A. Arbib, A. Billard, M. Iacoboni, and E. Oztop. Synthetic brain imaging: grasping, mirror neurons and imitation. *Neural Networks*, 13(8-9):975–997, 2000.

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011.

F. M. Bass, E. A. Pessemier, and D. R. Lehmann. An experimental study of relationships between attitudes, brand preference, and choice. *Behavioral Science*, 17(6):532–541, 1972.

J. Bateman, M. Beetz, D. Beßler, A. K. Bozcuoglu, and M. Pomarlan. Heterogeneous ontologies and hybrid reasoning for service robotics: The ease framework. In *Third Iberian Robotics Conference*, ROBOT '17, Sevilla, Spain, 2017.

S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneijder, and L. A. Stein. OWL Web Ontology Language Reference. Recommendation, World Wide Web Consortium (W3C), February10 2004. See `http://www.w3.org/TR/owl-ref/`.

M. Beetz. Structured reactive controllers: controlling robots that perform everyday activity. In *Proceedings of the third annual conference on Autonomous Agents*, pages 228–235. ACM, 1999.

M. Beetz. *Concurrent Reactive Plans: Anticipating and Forestalling Execution Failures*, volume 1772 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, 2000.

M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth. Robotic roommates making pancakes. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 529–536. IEEE, 2011a.

M. Beetz, U. Klank, A. Maldonado, D. Pangercic, and T. Rühr. Robotic roommates making pancakes - look into perception-manipulation loop. In *IEEE International Conference on Robotics and Automation (ICRA), Workshop on Mobile Manipulation: Integrating Perception and Manipulation*, pages 529–536, May, 9–13 2011b.

M. Beetz, D. Jain, L. Mosenlechner, M. Tenorth, L. Kunze, N. Blodow, and D. Pangercic. Cognition-enabled autonomous robot control for the realization of home chore task intelligence. *Proceedings of the IEEE*, 100(8):2454–2471, 2012a.

M. Beetz, L. Mösenlechner, M. Tenorth, and T. Rühr. Cram – a cognitive robot abstract machine. In *5th International Conference on Cognitive Systems (CogSys 2012)*, 2012b.

M. Beetz, M. Tenorth, and J. Winkler. Open-EASE – a knowledge processing service for robots and robotics/ai researchers. In *IEEE International Conference on Robotics*

and Automation (ICRA), Seattle, Washington, USA, 2015. Finalist for the Best Cognitive Robotics Paper Award.

M. Beetz, D. Beßler, J. Winkler, J.-H. Worch, F. Balint-Benczedi, G. Bartels, A. Billard, A. K. Bozcuoglu, Z. Fang, N. Figueroa, A. Haidu, H. Langer, A. Maldonado, A.-L. Pais, M. Tenorth, and T. Wiedemeyer. Open robotics research using web-based knowledge services. In *International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016. IEEE.

M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. K. Bozcuoglu, and G. Bartels. Knowrob 2.0 – a 2nd generation knowledge processing framework for cognition-enabled robotic agents. In *International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018. IEEE.

G. A. Bekey, H. Liu, R. Tomovic, and W. J. Karplus. Knowledge-based control of grasping in robot hands using heuristics from human motor skills. *IEEE Transactions on Robotics and Automation*, 9(6):709–722, 1993.

H. R. Beom and H. S. Cho. A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. *IEEE transactions on Systems, Man, and Cybernetics*, 25(3):464–477, 1995.

D. Beßler, A. K. Bozcuoglu, and M. Beetz. Information System for Storage, Management and Usage for Embodied Intelligent Agents. In R. Drechsler and C. Große, editors, *Information Storage from a multi-disciplinary perspective*. Springer, 2019. in press.

A. Bierbaum, M. Rambow, T. Asfour, and R. Dillmann. Grasp affordances from multi-fingered tactile exploration using dynamic potential fields. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 168–174. IEEE, 2009.

A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer, 2008.

B. Bocsi, L. Csató, and J. Peters. Alignment-based transfer learning for robot models. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–7. IEEE, 2013.

A. K. Bozcuoglu and M. Beetz. A cloud service for robotic mental simulations. In *International Conference on Robotics and Automation (ICRA)*, Singapore, 2017. IEEE.

A. K. Bozcuoğlu and E. Şahin. Traversability on a simple humanoid: What did i just trip over? In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 701–706. IEEE, 2011.

A. K. Bozcuoglu, D. Beßler, and M. Beetz. Rendering semantically-annotated experiment videos out of robot memories. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Seoul, Korea, November 2–5 2015a.

A. K. Bozcuoglu, F. Yazdani, D. Beßler, B. Togorean, and M. Beetz. Reasoning on communication between agents in a human-robot rescue team. In *Towards Intelligent Social Robots – Current Advances in Cognitive Robotics*, Seoul, Korea, 2015b.

A. K. Bozcuoglu, G. Kazhoyan, Y. Furuta, S. Stelter, M. Beetz, K. Okada, and M. Inaba. The exchange of knowledge using cloud robotics. *Robotics and Automation Letters*, 3(2):1072–1079, April 2018a. doi: 10.1109/LRA.2018.2794626.

A. K. Bozcuoglu, G. Kazhoyan, Y. Furuta, S. Stelter, M. Beetz, K. Okada, and M. Inaba. The exchange of knowledge using cloud robotics. In *International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018b. IEEE. Finalist for the Best Cognitive Robotics Paper Award, Finalist for the Best Service Robotics Paper Award.

A. K. Bozcuoglu, Y. Furuta, K. Okada, M. Beetz, and M. Inaba. Continuous modeling of affordances in a symbolic knowledge base. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Montreal, Canada, 2019a. IEEE. Submitted.

A. K. Bozcuoglu, S. Jung, J. Kim, A. Heidu, B.-T. Zhang, and M. Beetz. Crowdsourcing big data using virtual reality and cloud roboticss. *Robotics and Automation Letters*, 2019b. submitted.

S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

Care-O-bot 3 (2011). Care-o-bot 3. `http://www.care-o-bot.de/en/care-o-bot-3.html`, 2017. Accessed: 2017-06-11.

N. L. Cassimatis, J. G. Trafton, M. D. Bugajska, and A. C. Schultz. Integrating cognition, perception and action through mental simulation in robots. *Robotics and Autonomous Systems*, 49(1):13–23, 2004.

S. Chernova and M. Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 233. ACM, 2007.

L. Contreras, H. Yokoyama, and H. Okada. Multimodal feedback for active robot-object interaction. In *Towards Robots that Exhibit Manipulation Intelligence (IROS 2018 Workshop)*, Madrid, Spain, 2018.

R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater. Learning grasp affordance densities. *Paladyn*, 2(1):1, 2011.

A. G. Di Nuovo, D. Marocco, S. Di Nuovo, and A. Cangelosi. Autonomous learning in humanoid robotics through mental imagery. *Neural Networks*, 41:147–155, 2013.

D. Druckman, R. Bjork, C. Performance, C. Education, D. Education, and N. Council. *In the Mind's Eye: Enhancing Human Performance*. National Academies Press, 1992. ISBN 9780309563093. URL `https://books.google.de/books?id=COMJvuIH9CEC`.

A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust rgb-d object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 681–687. IEEE, 2015.

I. Fette. The websocket protocol. Technical report, Internet Engineering Task Force (IETF), 2011.

R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

C. Finn, S. Levine, and P. Abbeel. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. In *ICML*, pages 49–58, 2016.

C. Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, 2017a.

C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-Shot Visual Imitation Learning via Meta-Learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 357–368, 2017b. ISBN 9781510829008.

H. Gardner. *Frames of mind: The theory of multiple intelligences*. Hachette UK, 2011.

J. J. Gibson. *The theory of affordances*, chapter 8, pages 127–143. Psychology Press, new ed edition, Sept. 1986. ISBN 0898599598. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0898599598`.

M. A. Gluck, E. Mercado, and C. E. Myers. *Learning and memory : from brain to behavior*. Worth Publishers, New York, second edition, 2011.

A. Haidu, D. Beßler, A. K. Bozcuoglu, and M. Beetz. Knowrob-sim: Game engine-enabled knowledge processing for cognition-enabled robot control. In *International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018. IEEE.

R. W. Hamilton, D. V. Thompson, Z. G. Arens, S. J. Blanchard, G. Häubl, P. Kannan, U. Khan, D. R. Lehmann, M. G. Meloy, N. J. Roese, et al. Consumer substitution decisions: an integrative framework. *Marketing Letters*, 25(3):305–317, 2014.

M. Hegarty. Mechanical reasoning by mental simulation. *Trends in cognitive sciences*, 8(6):280–285, 2004.

F. Heintz, J. Kvarnström, and P. Doherty. Bridging the sense-reasoning gap: Dyknow–stream-based middleware for knowledge processing. *Advanced Engineering Informatics*, 24(1):14–26, 2010.

C. M. Heyes and B. G. Galef Jr. *Social learning in animals: the roots of culture*. Elsevier, 1996.

D. M. Hilbert and D. F. Redmiles. Extracting usability information from user interface events. *ACM Computing Surveys*, 32(4):384–421, Dec. 2000. ISSN 0360-0300. doi: 10.1145/371578.371593. URL `http://doi.acm.org/10.1145/371578.371593`.

J. Ho and S. Ermon. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems 29*, pages 4565–4573, 2016. ISBN 0045-7906. doi: 10.1016/j.compeleceng.2013.11.024.

F. Hoffmann and G. Pfister. Evolutionary design of a fuzzy knowledge base for a mobile robot. *International Journal of Approximate Reasoning*, 17(4):447–469, 1997.

J. Hollerbach, M. Mason, and H. Christensen. A Roadmap for US Robotics – From Internet to Robotics. Technical report, Computing Community Consortium (CCC), 2009.

M. Horstmann. Development and implementation of a security concept for a web-based robot knowledge service. Bachelor's thesis, University of Bremen, Bremen, Germany, June 2015.

J. Huckaby and H. I. Christensen. Toward a knowledge transfer framework for process abstraction in manufacturing robotics. In *ICML Workshop on Theoretically Grounded Transfer Learning*, 2013.

J. O. Huckaby. *Knowledge transfer in robot manipulation tasks*. PhD thesis, Georgia Institute of Technology, 2014.

F. F. Ingrand, R. Chatila, R. Alami, and F. Robert. Prs: A high level supervision and control language for autonomous mobile robots. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pages 43–49. IEEE, 1996.

I. Jagielska, C. Matthews, and T. Whitfort. An investigation into the application of neural networks, fuzzy logic, genetic algorithms, and rough sets to automated knowledge acquisition for classification problems. *Neurocomputing*, 24(1-3): 37–54, 1999.

W. James. The principles of. *Psychology*, 2:94, 1890.

M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal. Learning locomotion over rough terrain using terrain templates. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 167–172, Oct 2009. doi: 10.1109/IROS.2009.5354701.

H. B. Kappes and C. K. Morewedge. Mental simulation as substitute for experience. *Social and Personality Psychology Compass*, 10(7):405–420, 2016.

S. Karapinar, D. Altan, and S. Sariel-Talay. A robust planning framework for cognitive robots. In *Proceedings of the AAAI-12 workshop on cognitive robotics (CogRob)*, 2012.

B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg. Cloud-based robot grasping with the google object recognition engine. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

T. Keller, P. Eyerich, and B. Nebel. Task planning for an autonomous service robot. In *Towards Service Robots for Everyday Environments*, pages 117–124. Springer, 2012.

W. G. Kennedy, M. D. Bugajska, W. Adams, A. C. Schultz, and J. G. Trafton. Incorporating mental simulation for a more effective robotic teammate. In *AAAI*, pages 1300–1305, 2008.

S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

J. Kober and J. Peters. Reinforcement learning in robotics: A survey. In *Reinforcement Learning*, pages 579–610. Springer, 2012.

J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154, 2004. doi: 10.1109/IROS.2004.1389727.

P. Kormushev, B. Ugurlu, S. Calinon, N. G. Tsagarakis, and D. G. Caldwell. Bipedal walking energy minimization by reinforcement learning with evolving policy parameterization. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 318–324. IEEE, 2011.

J. Kuffner. Cloud-enabled humanoid robots. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, Nashville TN, United States, Dec.*, 2010.

Y. Kuniyoshi, R. Fukano, T. Otani, T. Kobayashi, and N. Otsu. Haptic detection of object affordances by a multi-fingered robot hand. *International Journal of Humanoid Robotics*, 2(04):415–435, 2005.

L. Kunze, T. Roehm, and M. Beetz. Towards semantic robot description languages. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5589–5595, Shanghai, China, May, 9–13 2011.

J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, Sept. 1987. ISSN 0004-3702. doi: 10.1016/0004-3702(87)90050-6. URL `http://dx.doi.org/10.1016/0004-3702(87)90050-6`.

P. Langley, D. Choi, and S. Rogers. Acquisition of hierarchical reactive skills in a unified cognitive architecture. *Cognitive Systems Research*, 10(4):316–332, 2009.

S. Lemaignan, A. Siadat, J.-Y. Dantan, and A. Semenenko. Mason: A proposal for an ontology of manufacturing domain. In *Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on*, pages 195–200. IEEE, 2006.

G. Levine and G. DeJong. Explanation-based acquisition of planning operators. In *ICAPS06, the Sixteenth International Conference on Automated Planning and Scheduling*, pages 152–161, 2006.

G. Lisca, D. Nyga, F. Bálint-Benczédi, H. Langer, and M. Beetz. Towards Robots Conducting Chemical Experiments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7354110`.

J. MacGlashan. The brown-umbc reinforcement learning and planning. `http://burlap.cs.brown.edu`, 2016. Accessed: 2016-09-10.

A. Mandlekar, Y. Zhu, A. Garg, Z. Silverstein, S. Darwish, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *CoRL*, 2018.

L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty, A. Kleiner, V. Lippiello, A. Finzi, B. Siciliano, A. Sala, and N. Tomatis. The sherpa project: smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, College Station, Texas, USA, Nov. 5-8 2012.

T. Matsui and M. Inaba. Euslisp: An object-based implementation of lisp. *Journal of Information Processing*, 13(3):327–338, 1990.

D. Mcdermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL - The Planning Domain Definition Language. Technical report, CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.212`.

B. McLaughlin. " intentional" and" incidental" learning in human subjects: The role of instructions to learn and motivation. *Psychological Bulletin*, 63(5):359, 1965.

D. Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, 2014.

H. Messerschmidt, J. Winkler, A. K. Bozcuoglu, and M. Beetz. Experience based task and environment specification for plan-based robotic agents. In *27. Workshop on Planen, Scheduling und Konfigurieren, Entwerfen*, Koblenz, Germany, September 16 2013.

J. Michels, A. Saxena, and A. Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 593–600. ACM, 2005.

D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *Robotics: Science and Systems (RSS)*. IEEE, 2014.

G. Mohanarajah, D. Hunziker, M. Waibel, and R. D'Andrea. Rapyuta: A cloud robotics platform. *IEEE Transactions on Automation Science and Engineering*, February 2014.

R. Moratz and T. Tenbrink. Affordance-based human-robot interaction. *Towards Affordance-Based Robot Control*, pages 63–76, 2008.

L. Mösenlechner and M. Beetz. Fast temporal projection using accurate physics-based geometric reasoning. In *2013 IEEE International Conference on Robotics and Automation*, pages 1821–1827, May 2013. doi: 10.1109/ICRA.2013.6630817.

A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming Exploration in Reinforcement Learning with Demonstrations. In *International Conference on Robotics and Automation*, 2018.

A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, volume 0, pages 663–670, 2000. ISBN 1-55860-707-2. doi: 10.2460/ajvr.67.2.323.

A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer, 2006.

D. Nguyen-Tuong, M. Seeger, and J. Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

T. Niemueller, G. Lakemeyer, and S. S. Srinivasa. A Generic Robot Database and its Application in Fault Analysis and Performance Evaluation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012*, Vilamoura, Algarve, Portugal, 2012. IEEE, IEEE/RAS.

N. J. Nilsson. Shakey the robot. Technical report, SRI INTERNATIONAL MENLO PARK CA, 1984.

D. Nyga, M. Picklum, S. Koralewski, and M. Beetz. Instruction Completion through Instance-based Learning and Semantic Analogical Reasoning. In *International Conference on Robotics and Automation (ICRA)*, Singapore, 2017. IEEE.

T. Ogura, K. Okada, and M. Inaba. Cutting vegetables tasks by humanoid robot using vision and force sensors. In *The 24th Annual Conference on Robotics Society of Japan*, 2006.

K. Okada, Y. Kino, M. Inaba, and H. Inoue. Visually-based humanoid remote control system under operator's assistance and its application to object manipulation. In *Proceedings of Third IEEE International Conference on Humanoid Robots*, 2003.

A. L. Pais, B. D. Argall, and A. G. Billard. Assessing interaction dynamics in the context of robot programming by demonstration. *International Journal of Social Robotics*, 5(4):477–490, 2013.

D. Pangercic, M. Tenorth, B. Pitzer, and M. Beetz. Semantic object maps for robotic housework - representation, acquisition and use. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October, 7–12 2012.

W. T. Park. The sri robot programming system (rps). *Technical Paper Society of Mechanical Engineers*, pages 22–41, 1983.

D. Paulius, Y. Huang, R. Milton, W. D. Buchanan, J. Sam, and Y. Sun. Functional object-oriented network for manipulation learning. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2655–2662. IEEE, 2016.

D. N. Perkins and G. Salomon. Are cognitive skills context-bound? *Educational researcher*, 18(1):16–25, 1989.

L. B. Pham and S. E. Taylor. From Thought to Action: Effects of Process-Versus Outcome-Based Mental Simulations on Performance. *Personality and Social Psychology Bulletin*, 25(2):250–260, Feb. 1999. doi: 10.1177/0146167299025002010. URL `http://dx.doi.org/10.1177/0146167299025002010`.

D. A. Phillips, J. P. Shonkoff, et al. *From neurons to neighborhoods: The science of early childhood development*. National Academies Press, 2000.

L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3406–3413. IEEE, 2016.

D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 305–313. Morgan-Kaufmann, 1989.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source Robot Operating System. In *IEEE International Conference on Robotics and Automation, 2009*, Kobe, Japan, May 12–17 2009a.

M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009b.

A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Robotics: Science and Systems (RSS)*, 2018.

K. Ramirez-Amaro, M. Beetz, and G. Cheng. Transferring skills to humanoid robots by extracting semantic representations from observations of human activities. *Artificial Intelligence*, 247:95–118, 2017.

L. Riazuelo, M. Tenorth, D. D. Marco, M. Salas, D. Gálvez-López, L. Mösenlechner, L. Kunze, M. Beetz, J. D. Tardós, L. Montano, and J. M. M. Montiel. RoboEarth Semantic Mapping: A Cloud Enabled Knowledge-Based Approach. *IEEE Transactions on Automation Science and Engineering (T-ASE)*, 2015. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7015601`.

S. Ross, G. Gordon, and D. Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

J. Rothfuss, F. Ferreira, E. E. Aksoy, Y. Zhou, and T. Asfour. Deep episodic memory: Encoding, recalling, and predicting episodic experiences for robot action execution. *arXiv preprint arXiv:1801.04134*, 2018.

P. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, Nov. 1987. ISSN 0377-0427. doi: 10.1016/0377-0427(87)90125-7. URL `http://dx.doi.org/10.1016/0377-0427(87)90125-7`.

D. Roy, K.-Y. Hsiao, and N. Mavridis. Mental imagery for a conversational robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(3): 1374–1383, 2004.

G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering, 1994.

R. B. Rusu, Z. C. Marton, N. Blodow, A. Holzbach, and M. Beetz. Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen

Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 11-15 2009.

E. D. Sacerdoti. A structure for plans and behavior. Technical report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1975.

O. Saha and P. Dasgupta. A comprehensive survey of recent trends in cloud robotics architectures and applications. *Robotics*, 7(3):47, 2018.

M. Saito, H. Chen, K. Okada, M. Inaba, L. Kunze, and M. Beetz. Semantic object search in large-scale indoor environments. In *Proceedings of IROS 2012 Workshop on active Semantic Perception and Object Search in the Real World*. IEEE, 2011.

A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, and H. S. Koppula. Robo brain: Large-scale knowledge engine for robots. Technical report, arXiv, 2014. http://arxiv.org/abs/1412.0691.

C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, T. Kramer, and E. Miguelanez. An ieee standard ontology for robotics and automation. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1337–1342. IEEE, 2012.

S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3):123–158, 1996.

J. A. Sommerville, A. L. Woodward, and A. Needham. Action experience alters 3-month-old infants' perception of others' actions. *Cognition*, 96(1):B1–B11, 2005.

F. Stulp, A. Fedrizzi, L. Mösenlechner, and M. Beetz. Learning and Reasoning with Action-Related Places for Robust Mobile Manipulation. *Journal of Artificial Intelligence Research (JAIR)*, 43:1–42, 2012.

J. Sung, B. Selman, and A. Saxena. Synthesizing manipulation sequences for under-specified tasks using unrolled markov random fields. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2970–2977, Sept 2014. doi: 10.1109/IROS.2014.6942972.

J. Sung, S. H. Jin, and A. Saxena. Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *Robotics Research*, pages 701–720. Springer, 2018.

R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

S. E. Taylor, L. B. Pham, I. D. Rivkin, and D. A. Armor. Harnessing the imagination: Mental simulation, self-regulation, and coping. *American psychologist*, 53(4):429, 1998.

M. Tenorth and M. Beetz. KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. *Int. Journal of Robotics Research*, 32(5):566 – 590, April 2013. URL `http://ijr.sagepub.com/content/32/5/566.short`.

M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz. The RoboEarth language: Representing and Exchanging Knowledge about Actions, Objects, and Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 14–18 2012. Best Cognitive Robotics Paper Award.

M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz. Representation and Exchange of Knowledge about Actions, Objects, and Environments in the RoboEarth Framework. *IEEE Transactions on Automation Science and Engineering (T-ASE)*, 10 (3):643–651, 2013. ISSN 1545-5955. doi: 10.1109/TASE.2013.2244883. Best Paper Award Finalist.

A. L. Thomaz and M. Cakmak. Learning about objects with human teachers. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 15–22. ACM, 2009.

S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, et al. Minerva: A second-generation museum tour-guide robot. In *Robotics and automation, 1999. Proceedings. 1999 IEEE international conference on*, volume 3. IEEE, 1999.

E. Tulving. Episodic memory: from mind to brain. *Annual review of psychology*, 53 (1):1–25, 2002.

E. Tulving et al. Episodic and semantic memory. *Organization of memory*, 1:381–403, 1972.

E. Uğur and E. Şahin. Traversability: A case study for learning and perceiving affordances in robots. *Adaptive Behavior*, 18(3-4):258–284, 2010.

E. Ugur, E. Şahin, and E. Oztop. Self-discovery of motor primitives and learning grasp affordances. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3260–3267. IEEE, 2012.

K. M. Varadarajan and M. Vincze. Afrob: The affordance network ontology for robots. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1343–1350. IEEE, 2012.

R. Vuga, E. E. Aksoy, F. Wörgötter, and A. Ude. Probabilistic semantic models for manipulation action representation and extraction. *Robotics and Autonomous Systems*, 65:40 – 56, 2015. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2014.11.012. URL `http://www.sciencedirect.com/science/article/pii/S0921889014002851`.

M. Waibel, M. Beetz, J. Civera, R. d'Andrea, J. Elfring, D. Galvez-Lopez, K. Häussermann, R. Janssen, J. Montiel, A. Perzylo, et al. Roboearth. *IEEE Robotics & Automation Magazine*, 18(2):69–82, 2011.

Y. Wakita, S. Hirai, T. Hori, R. Takada, and M. Kakikura. Realization of safety in a coexistent robotic system by information sharing. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 3474–3479. IEEE, 1998.

J. Wielemaker. An overview of the SWI-Prolog programming environment. In F. Mesnard and A. Serebenik, editors, *Proceedings of the 13th International Workshop on Logic Programming Environments*, pages 1–16, Heverlee, Belgium, december 2003. Katholieke Universiteit Leuven. CW 371.

Willow Garage Personal Robot 2 (2011). Ros/robots/pr2. `http://wiki.ros.org/Robots/PR2`, 2017. Accessed: 2017-06-11.

J. Winkler, M. Tenorth, A. K. Bozcuoglu, and M. Beetz. CRAMm – memories for robots performing everyday manipulation activities. *Advances in Cognitive Systems*, 3:47–66, 2014.

J. Winkler, A. K. Bozcuoglu, M. Pomarlan, and M. Beetz. Task parametrization through multi-modal analysis of robot experiences (extended abstract). In *Proceedings of the 2017 International Conference on Autonomous Agents*, AAMAS '17, 2017.

R. Wood, P. Baxter, and T. Belpaeme. A review of long-term memory in natural and synthetic systems. *Adaptive Behavior*, 20(2):81–103, 2012.

K. A. Wyrobek, E. H. Berger, H. F. M. V. der Loos, and J. K. Salisbury. Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In *2008 IEEE International Conference on Robotics and Automation*, pages 2165–2170, May 2008. doi: 10.1109/ROBOT.2008.4543527.

F. Yazdani, G. Kazhoyan, A. K. Bozcuoglu, A. Haidu, F. Balint-Benczedi, D. Beßler, M. Pomarlan, and M. Beetz. Cognition-enabled framework for mixed human-robot rescue team. In *International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018. IEEE.

F. Yazdani, S. Blumenthal, N. Huebel, A. K. Bozcuoglu, M. Beetz, and H. Bruyninckx. Query-based integration of heterogeneous knowledge bases for search and rescue tasks. *Robotics and Autonomous Systems (RAS)*, Special Issue on Semantic Policy and Action Representations for Autonomous Robots, 2019.

T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine. One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning. In *Robotics: Science and Systems (RSS)*, feb 2018.

P. Zech, S. Haller, S. R. Lakani, B. Ridge, E. Ugur, and J. Piater. Computational models of affordance in robotics: a taxonomy and systematic classification. *Adaptive Behavior*, 25(5):235–271, 2017. doi: 10.1177/1059712317726357. URL `https://doi.org/10.1177/1059712317726357`.

F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*, 2015.